Design and Implementation of an Educational CPU

Rex N. Fisher

University of Idaho

April 7, 1997

(Revised on Feb 3, 2000)

Contents

List of Figures

Abstract

Reviews of current textbooks about computer fundamentals, and actual classroom experiences with students, show a need for a working circuit that actually performs the basic functions illustrated by simple, educational CPU models. Several CPU models in current textbooks were evaluated for implementation, but each of them had at least one feature that eliminated it from consideration. A new CPU model was developed that has been constructed with 23 common ICs. It has an 8-bit data bus, 8-bit address bus, ALU, 2 data registers, 1 condition register, and a microprogrammed control unit. The instruction set is complete, orthogonal, and includes 16 types of instructions: 2 I/O, 4 program control (including conditional branching), 4 data transfer, 3 arithmetic, and 3 logical instructions. It supports 4 addressing modes: direct, indirect, register, and immediate. All information required to duplicate it, such as schematics and microprogram listings, are included in the appendixes.

Summary

Textbooks for introductory courses about computer organization or computer architecture often contain simple CPU models (usually block diagrams) that illustrate the basic operations common to most CPUs. But, the details of how these functional blocks actually work are rarely discussed. Many students have a background in digital circuit theory. They not only *can* understand the circuit details, but often *want* to understand them. Classroom experience with students in the TAC/ABET accredited Electronics Engineering Technology program at Ricks College prompted the development of a working CPU that actually implements the operations described by such a CPU model.

Four "paper" CPU models used in other college classrooms were evaluated for hardware implementation. They were all rejected. Those that were simple enough to implement with about 20 ICs were functionally inadequate to illustrate many principles. The more complete models required 40 or 50 ICs to build.

A new CPU model, called the P8, was developed to combine simplicity with completeness. The programmer's model is shown in Figure S.1. It has three 8-bit registers that are visible to users: the instruction pointer (IP), the A register, and the R register. A 1-bit zero condition register (Z) is also included. Because the IP is 8 bits wide, only 256 memory locations may be addressed. There is also a separate space of 256 I/O port addresses.

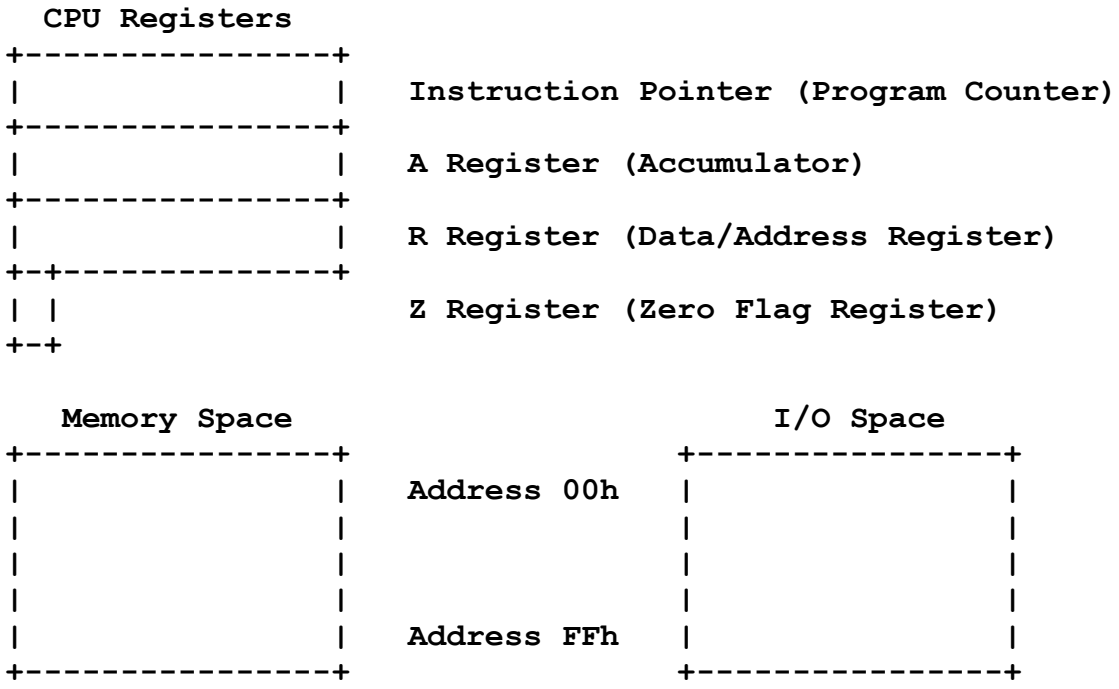


Figure S.1: Programmer's Model of MP8 CPU.

The P8 CPU has a complete and orthogonal instruction set. The selection of instructions was based on published benchmarks of the instructions used most frequently by the Intel 8086 microprocessor. There are 16 types of P8 instructions. They fall into five categories.

1. Input / Output. These instructions transfer data between the accumulator and external I/O devices.

   IN = Read Input Port
   OUT = Write Output Port

2. Program Control. These instructions change the sequence of program execution. They are often called branch instructions.

   JMP = Unconditional Jump
   JNZ = Jump If Not Zero (Conditional Jump)
   JZ = Jump If Zero (Conditional Jump)
   CMP = Compare (Sets / Resets Zero Bit For Conditional Jumps)

3. Data Transfer. These instructions cause data in one location (either the internal registers or external memory) to be copied to another location.

    LDA = Load A Register
    LDR = Load R Register
    STA = Store A Register
    STR = Store R Register

4. Arithmetic. These instructions perform numerical operations on data. (Floating point operations are not supported.)

    ADD = Add To A Register
    SUB = Subtract From A Register
    DEC = Decrement

5. Logical. These instructions perform Boolean operations on data, including bit shifting.

    OR = Or With A Register
    INV = Invert & Move To A Register
    SHL = Shift Left & Move To A Register

Four addressing modes are supported. They were also selected as a result of published benchmarks compiled for the 8086 microprocessor.

1. Direct
2. Indirect
3. Register
4. Immediate

A functional block diagram is shown in Figure 2. It has been implemented with 23 ICs:

```
C    1 - 74LS74     Dual D Flip Flop
C    3 - 74LS163    4-Bit Binary Counter
C    2 - 74LS181    4-Bit ALU
C    3 - 74LS244    Octal 3-State Buffer
C    1 - 74LS273    Octal D Flip Flop w/ Clear
C    6 - 74LS374    Octal D Flip Flop w/ 3-State Output
C    3 - GAL16V8 Programmable Array Logic
C    4 - 2764  8K X 8 PROM
```

The CPU's datapath has an 8-bit internal bus that conveys data between the registers. The A register is the accumulator. It is the destination for all ALU results. The R register can be used

for general data storage or as an address pointer for instructions that use indirect addressing. The data register (DR) reads, and writes to, the external data bus. The address register (AR) writes to the external address bus. The instruction pointer (IP) keeps track of where the next instruction in memory is located. The operand register (OR) is used to write operand addresses to AR. The instruction register (IR) holds the 8-bit opcode of the instruction being executed.

The control unit consists of the zero condition register (Z) and the block labeled "control logic". The control logic is a 31-bit control store that contains the microinstructions required to execute each of the P8 CPU's instructions. The correct microinstructions are selected with a combination of inputs from IR, Z, and the microinstruction pointer (MIP), which is a binary counter within the control logic block that sequences through the correct microinstructions for each instruction that is to be executed.

All information required to duplicate the P8 CPU, including schematics, PAL code, and microcode listings, can be found in the appendixes.

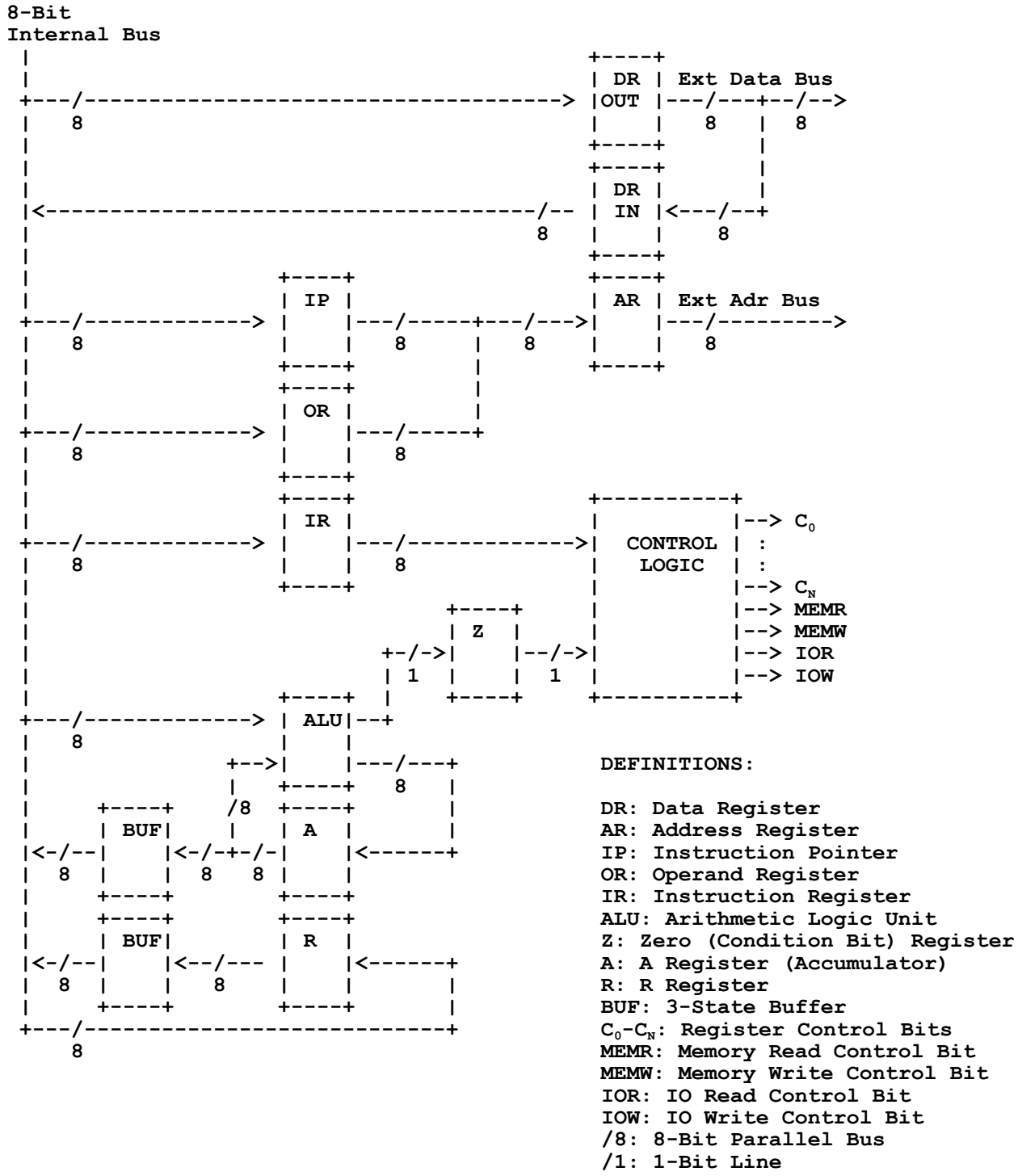


Figure S.2: Functional Block Diagram of MP8 CPU

1.0 Introduction

Block diagrams of simple CPUs have been used in beginning computer courses for decades, because it is important to visualize how a CPU functions. To meet this need, many textbook authors have devised simple CPUs at the block diagram level to illustrate how instructions are executed and data are manipulated. Omitting many of the circuit details allows an overall understanding that is usually sufficient for students with little or no experience with digital circuits.

However, students in TAC/ABET accredited Electronics Engineering Technology programs, such as the one at Ricks College, have a more thorough background in digital circuit design. They are able to understand how instructions are decoded, what control signals are required for datapath operation, and how those control signals are generated. By examining this extra level of detail, students can better tie the new material to principles they have already learned.

This report describes the development and operation of the P8 CPU. It is a working 8-bit CPU designed and built with standard memory devices and TTL logic components. It extends a student's understanding of computer operation to a deeper level than can be accomplished with only a block diagram CPU model.

Four elements of this project will be explained: (1) the selection of a suitable CPU model; (2) the details of the instruction set, with its various operations and addressing modes; (3) the design of the actual datapath and control logic hardware; and (4) the operation of the CPU, along with some examples of how the hardware implements the instruction set.

2.0 Literature Review & Analysis

This section highlights the importance of using a working CPU model in Electronics Engineering Technology computer courses. It also describes the investigation of existing CPU models that were considered for hardware implementation, and the results of that research.

2.1 The Need for a Simple CPU Model that Actually Works

The two-year Electronics Engineering Technology program at Ricks College has several courses about microprocessors, microcontrollers and computer systems. A prerequisite for all of these courses is EET 151 (Digital Circuits). EET 151 covers combinational SSI circuits, as well as MSI circuits such as ALUs, PALs, and semiconductor memories. The treatment of sequential circuits includes flip flops, latches, registers, counters, and an introduction to state machines. Students in this class learn digital hardware design at a deeper level than do most two-year Computer Science or Vocational Technology students. Throughout the course the students learn that these are the basic building blocks of computers.

Textbooks for the subsequent computer courses, however, treat the internal circuitry of a CPU simply as a functional block diagram. The students are told that this block is a register and that block is an ALU, etc., but the bridge between the block diagram and the circuits studied in EET 151 is never completely built. Students often ask, "Can you show me where that functional block is in a real computer?" They want to make the connection back to the hardware they learned about in Digital Circuits.

Few of the CPU models found in these beginner-level computer textbooks explain how the CPU "knows" what instruction was fetched. Somehow a block called "control logic" sets up the ALU for the correct operation and loads all of the correct registers at the proper time. But, students in Electronics Engineering Technology want to know how this is accomplished.

By building a simple demonstration CPU that actually works, both levels of understanding can be satisfied. The block diagram can be used to learn the basic principles, and the detailed design of the functional CPU can tie the functional blocks back to actual circuits.

2.2 Initial CPU Model Specifications

Ten loosely defined CPU specifications were proposed for this project. The CPU had to be powerful enough to illustrate most of the common features in modern CPUs. It would also have to be simple enough to implement at the IC level within a short amount of time. Although some of these specifications were later modified or discarded, they were used to evaluate the existing CPU models and formed the design basis for the P8 CPU.

1. The CPU should be simple enough to be implemented with a "small" number of ICs (about 20) from the TTL logic family.

2. The control unit should support conditional branching.

3. Include a hardware interrupt.

4. Use an 8-bit datapath and 8-bit address bus to reduce complexity.

5. Incorporate a "small" number of internal data registers (about 3) to demonstrate how they function.

6. The ALU should implement an assortment of arithmetic and logical operations, in order to make the instruction set as complete as possible.

7. The instruction set should include "several" (3 or 4) popular addressing modes.

8. Include instructions that are usually found in real CPUs.

9. Implement simple stack operations.

10. No pipelining or other parallel processing techniques should be attempted. They would detract from the simple operation required for easy understanding, and would certainly conflict with goal # 1.

2.3 Alternative Computer Models

This section briefly describes four different CPU models that have been developed for use in college-level computer courses. They are identified in this report by the name of their respective developers. Each one will be compared to the design goals in the previous section. The features, and advantages and disadvantages of each will be identified. The suitability of each model for hardware implementation will be discussed.

2.3.1 Lynn CPU

This CPU model was developed by Doug Lynn for use in his EE 340 and EE 441 courses at the University of Idaho (Lynn, 1996). A diagram of the Lynn CPU is shown in Figure 2.1.

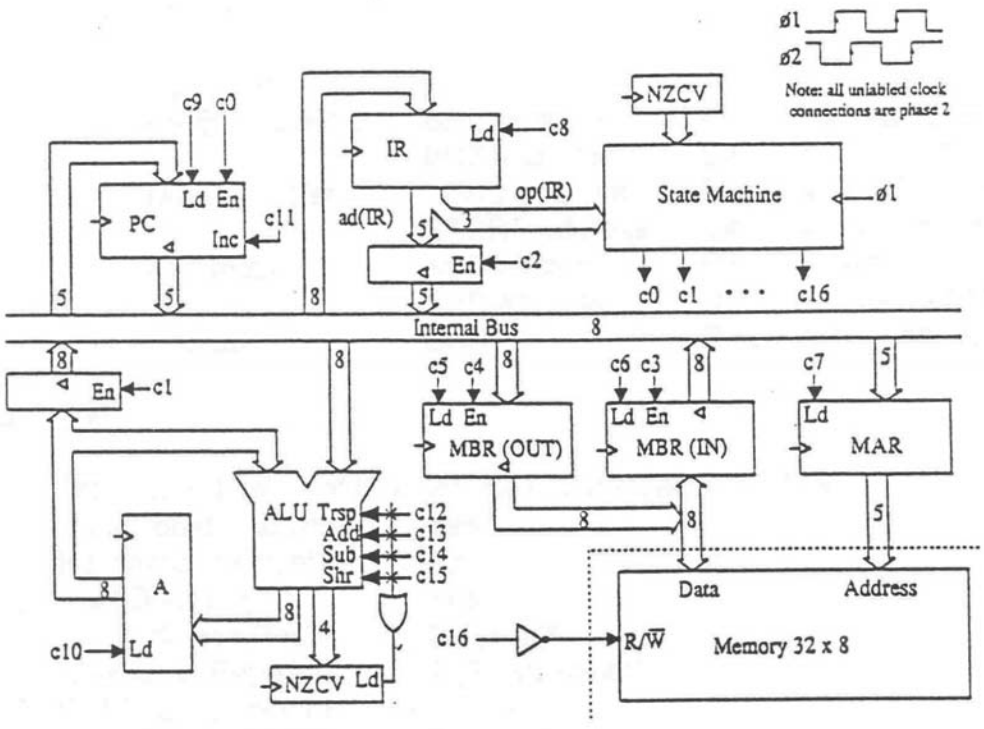


Figure 2.1: Functional Block Diagram of Lynn CPU (Lynn, 1996, EE 441 Handout).

Features

CPU Architecture: Accumulator-Based CPU

Data Bits: 8

Address Bits: 5

Internal Registers: 1 Accumulator (A)

Flags:
- Zero (Z)
- Carry (C)
- Negative (N)
- Overflow (V)

Instruction Set:

Input/Output:	None	--	--
Prog Control:	BRU addr	Dir	PC <--addr
	BRZ addr	Dir	If Z=1: PC <-- addr
Data Transfer:	LOAD add	Dir	A <-- MEM(addr)
	STORE addr	Dir	MEM(addr) <-- A
Arithmetic:	ADD addr	Dir	A <-- A + MEM(addr)
	SUB addr	Dir	A <-- A - MEM(addr)
Logical	SHR	Reg	A <-- 0 ## $[A]_{7..1}$

Evaluation

Estimated Number of ICs:	20 (12 + State Machine + Memory)
Conditional Branching?:	Yes
Hardware Interrupt?:	No
8-Bit Data Bus?:	Yes
8-Bit Address Bus?:	No (5 bits)
Number of Registers:	1 (Accumulator)
# of ALU Operations:	3
# of Addressing Modes:	2
Stack Operations?:	No
Pipelining?:	No

Decision

The CPU would be easy enough to build, but it does not have many features common to contemporary CPUs. Its instruction set is too limited and does not implement all of the hardware features. Different addressing modes cannot be

demonstrated effectively. The available memory is too small to run an actual program.

This CPU model would not be an acceptable candidate for implementation.

2.3.2 Streib CPU

William Streib presents this model in his book (Streib, 1997). See the block diagram in Figure 2.2.

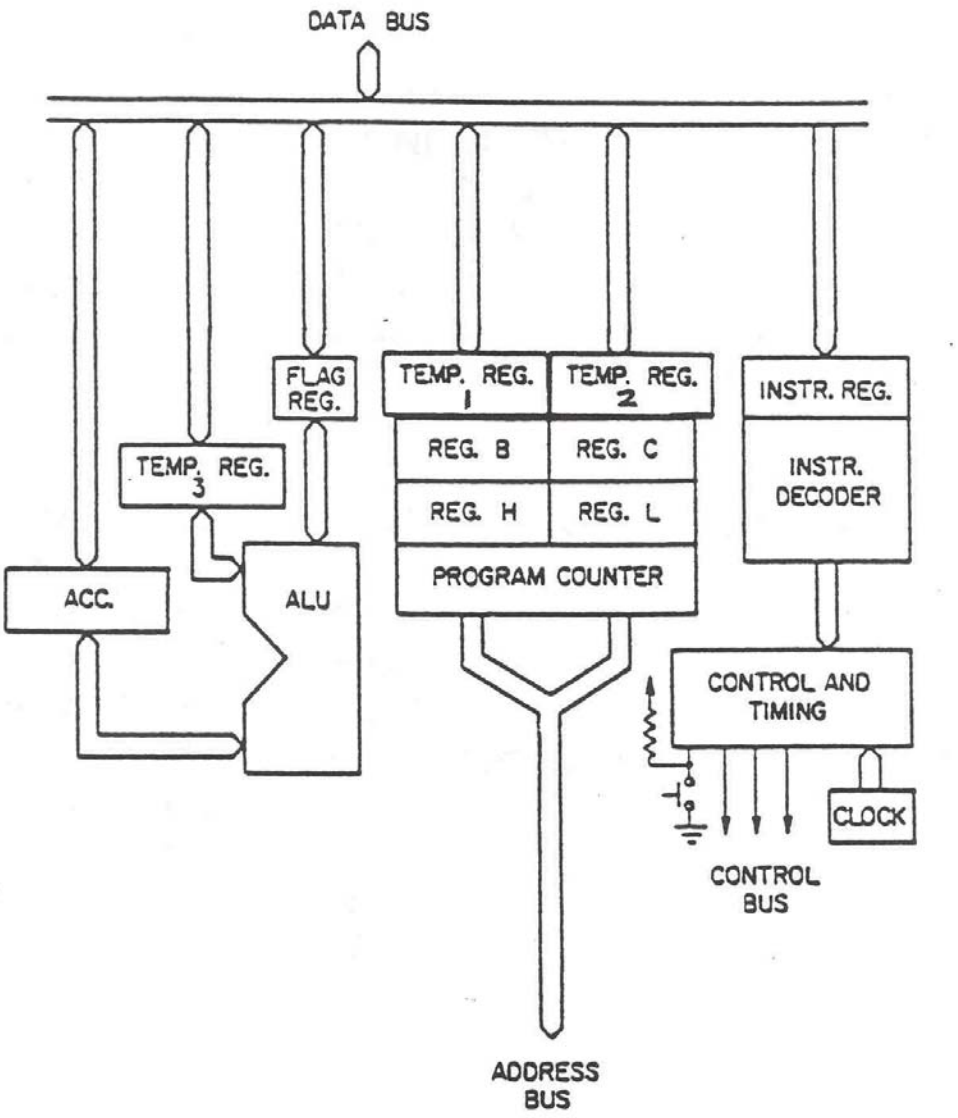


Figure 2.2: Functional Block Diagram of Streib CPU (Streib, 1997, p. 323).

Features

CPU Architecture: Accumulator-Based

Data Bits: 8

Address Bits: 16

Internal Registers: 1 Accumulator (A)
2 General Purpose Registers (B, C)
2 Special Purpose Registers (H, L)

Flag: Zero (Z)

Instruction Set:

Input/Output:	INPUT addr	Dir	A <-- PORT(addr)
Prog Control:	JUMP addr	Dir	PC <-- addr
	COND JUMP addr	Dir	If Z = 0: JUMP addr
Data Transfer:	LOAD A addr	Dir	A <-- MEM(addr)
	MOVE A, B	Reg	A <-- B
Arithmetic:	None	-	-
Logical:	AND data	Imm	A <-- A _ data

Evaluation

Estimated Number of ICs: 24 (14 + Decoder + Control & Timing + Memory)
Conditional Branching?: Yes
Hardware Interrupt?: No
8-Bit Data Bus?: Yes
8-Bit Address Bus?: No (16 bits)
Number of Registers: 5 (1 Accumulator + 2 GPs + 2 SPs)
# of ALU Operations: 1
# of Addressing Modes: 3
Stack Operations?: No
Pipelining?: No

Decision

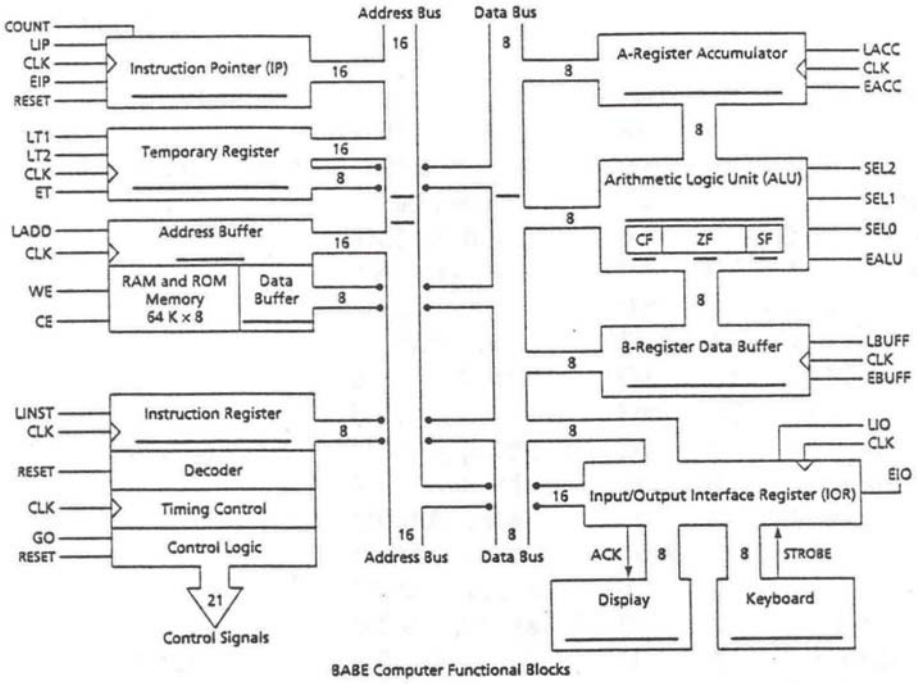
The CPU would be only slightly more difficult to build than the Lynn CPU. There is ample memory space. It has many hardware features common to contemporary CPUs, but its instruction set does not implement most of them. The instruction set uses three different addressing modes, but

is very weak functionally -- especially the available ALU operations.

This CPU model would not be an acceptable candidate for implementation.

2.3.3 Miller CPU

Michael Miller actually has his own name for this one -- Binary Architecture Basic Electronic (BABE) Computer (Miller, 1997).



BABE Computer Functional Blocks

Figure 2.3: Functional Block Diagram of Miller (BABE) CPU (Miller, 1997, p. 497).

Features

CPU Architecture: Accumulator-Based

Data Bits: 8

Address Bits: 16

Internal Registers:
1 Accumulator (A)
1 General Purpose Register (B)
1 I/O Register

Flags:
Zero (ZF)
Sign (SF)
Carry (CF)

Instruction Set:

Input/Output:	IN	Reg	A <-- I/O Register
	OUT	Reg	I/O Register <-- A
Prog Control:	JMP addr	Dir	IP <-- addr
	JZ addr	Dir	If Z = 1: JMP addr
	JNZ addr	Dir	If Z = 0: JMP addr
	JM addr	Dir	If S = 1: JMP addr
	JP addr	Dir	If S = 0: JMP addr
	JC addr	Dir	If C = 1: JMP addr
	JNC addr	Dir	If C = 0: JMP addr
	HALT	Imp	PC <-- PC
Data Transfer:	MOV A, [addr]	Dir	A <-- MEM(addr)
	MOV B, [addr]	Dir	B <-- MEM(addr)
	MOV A, data	Imm	A <-- data
	MOV B, data	Imm	B <-- data
	MOV [addr, A	Dir	MEM(addr) <-- A
	MOV [addr], A	Dir	MEM(addr) <-- A
Arithmetic:	ADD [addr]	Dir	A <-- A + MEM(addr)
	ADD data	Imm	A <-- A + data
	ADD B	Reg	A <-- A + B
	SUB [addr]	Dir	A <-- A - MEM(addr)
	SUB data	Imm	A <-- A - data
	SUB B	Reg	A <-- A - B
	DEC A	Reg	A <-- A - 1
	DEC B	Reg	B <-- B - 1
	INC A	Reg	A <-- A + 1
	INC B	Reg	B <-- B + 1
Logical:	SHL	Reg	A <-- $[A]_{6..0}$ ## 0
	SHR	Reg	A <-- 0 ## $[A]_{7..1}$

Evaluation

Estimated Number of ICs: 55 (40 + ALU* + Control + Memory)
Conditional Branching?: Yes
Hardware Interrupt?: No
8-Bit Data Bus?: Yes
8-Bit Address Bus?: No (16 bits)
Number of Registers: 3 (1 Accumulator + 1 GP + 1 SP)
# of ALU Operations: 10
# of Addressing Modes: 3
Stack Operations?: No
Pipelining?: No

*** There are no standard TTL ALU devices that implement a Shift Right function.**

Decision

There is ample memory space for even lengthy programs. It has many hardware features common to contemporary CPUs, and its instruction set implements most of them. Although the instruction set uses three different addressing modes, one of the most powerful -- indirect -- is not included. This is a surprising omission for an otherwise very complete CPU model. The biggest disadvantage of this one -- enough to eliminate it from consideration -- is that it would be incredibly time consuming to build!

This CPU model would not be an acceptable candidate for implementation.

2.3.4 McCalla CPU

This CPU model is Thomas McCalla's design (McCalla, 1992). See Figure 2.4 for his BC-8A Computer.

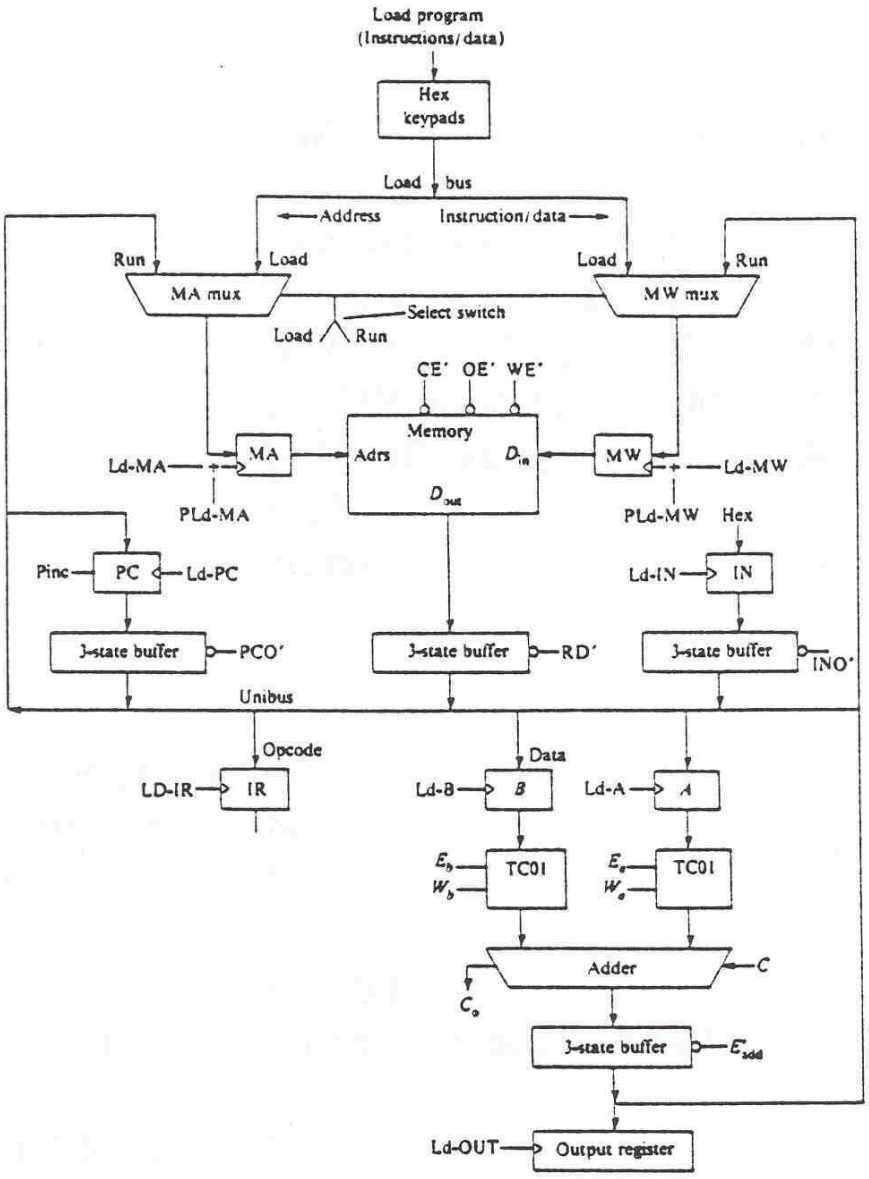


Figure 2.4: Functional Block Diagram of McCalla (BC-8A) CPU (McCalla, 1992, p. 673).

Features

CPU Architecture: Accumulator-Based CPU

Data Bits: 8

Address Bits: 5

Internal Registers: 1 Accumulator (A)
1 General Purpose Register (B)
1 I/O Register

Flags: None

Instruction Set:

Input/Output:	INP	Reg	A <-- PORT
	OUT	Reg	PORT <-- A
Prog Control:	JMP addr	Dir	PC <--addr
Data Transfer:	LDADIR addr	Dir	A <-- MEM(addr)
	STADIR addr	Dir	MEM(addr) <-- A
Arithmetic:	ADDIR addr	Dir	A <-- A + MEM(addr)
	SUBDIR addr	Dir	A <-- A - MEM(addr)
	CLA	Reg	A <-- 0
Logical	None	--	--

Evaluation

Estimated Number of ICs: 39
Conditional Branching?: No
Hardware Interrupt?: No
8-Bit Data Bus?: Yes
8-Bit Address Bus?: No (5 bits)
Number of Registers: 3 (Accumulator + 1 GP + 1 SP)
# of ALU Operations: 3
# of Addressing Modes: 2
Stack Operations?: No
Pipelining?: No

Decision

Almost no hardware design effort would be required for this CPU because the schematic is provided. See Figure 2.5. (That indicates that there are others who believe it is important to show how a simple CPU model can actually be implemented in hardware.) This CPU, however, has nearly twice as many ICs as the goal specifies. The Instruction

set is similar to the Lynn CPU and has the same inadequacies. It has only two addressing modes and no conditional branches at all. The available memory is also too small to run an actual program.

This CPU model would not be an acceptable candidate for implementation.

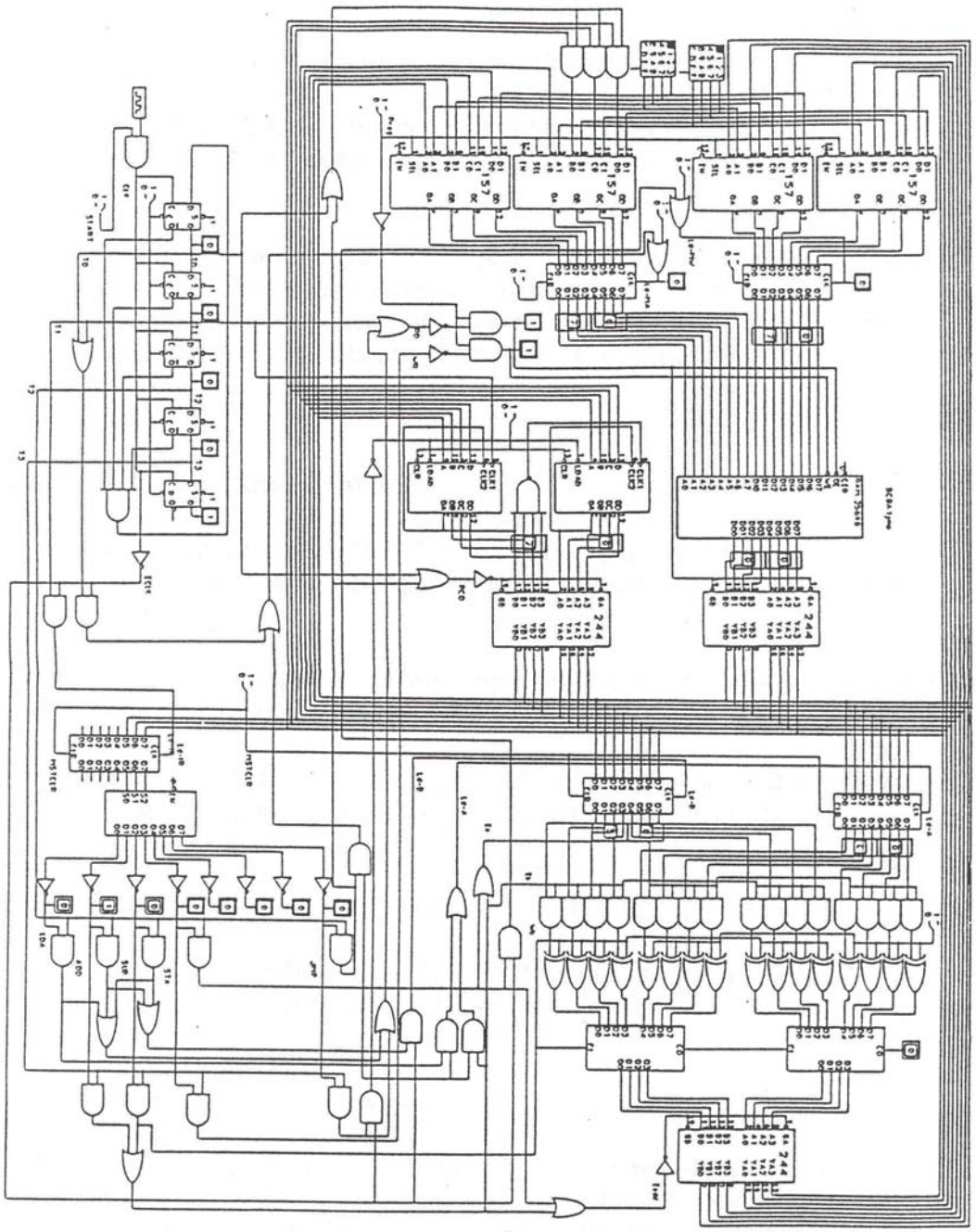


Figure 2.5: Schematic Diagram of McCalla CPU (McCalla, 1992, p.678-679).

2.4 Literature Review Conclusions

The fact that McCalla provided a complete schematic for his CPU and Streib illustrated a partial IC-level design show that there is a need for this project. None of the four CPU models reviewed, however, had all of the features listed in Section 2.2. This was expected, but each one also had at least one major drawback that totally eliminated it from consideration.

A new CPU model was developed for this project. Called the P8 (Project 8-bit) CPU, it incorporates most of the best features found in the other four CPUs. It is simple, yet has a complete, orthogonal, and reasonably powerful instruction set. Many addressing modes found in contemporary CPUs are supported.

2.5 Final CPU Model Specifications

During the development of the P8 CPU, some of the original design specifications were changed. Below are the ten original design goals, along with the actual results of those goals.

1. Original: The CPU should be simple enough to be implemented with a "small" number of ICs (about 20) from the TTL logic family.

   Final: The CPU is comprised of 23 ICs. Actually, only 22 are required for proper operation, but another was added to gain visibility into one of the registers.

2. Original: The control unit should support conditional branching.

   Final: "Jump If Zero" and "Jump If Not Zero" are supported. Zero is the only condition bit, but it is sufficient to illustrate the principle of conditional branching.

3. Original: Include a hardware interrupt.

   Final: This was not implemented. It would have required three additional ICs and a more complicated control unit. The additional complexity did not seem worth it. The fact that none of the other CPU designs incorporated a hardware interrupt indicates that other designers of educational CPUs agree with this decision.

4. **Original: Use an 8-bit datapath and 8-bit address bus to reduce complexity.**

   Final: No change.

5. **Original: Incorporate a "small" number of internal data registers (about 3) to demonstrate how they function.**

   Final: There are two data registers in the final design.

6. **Original: The ALU should implement an assortment of arithmetic and logical operations, in order to make the instruction set as complete as possible.**

   Final: The ALU performs the following operations:

   - **Compare**
   - **Add**
   - **Subtract**
   - **Decrement**
   - **Logical OR**
   - **Logical NOT (Invert)**
   - **Logical Shift Left**

7. **Original: The instruction set should include "several" (3 or 4) popular addressing modes.**

   Final: The following addressing modes are supported:

   - **Direct**
   - **Indirect**
   - **Register**
   - **Immediate**

8. **Original: Include instructions that are commonly found in real CPUs.**

   Final: Two-thirds of the most frequently used instructions are implemented.

9. **Original: Implement simple stack operations.**

   Final: This was not implemented. It would have required four additional ICs and a more complicated control unit. The additional complexity did not seem worth it. The fact that none of the other CPU designs incorporated stack operations indicates that other designers of educational CPUs agree with this decision.

10. Original: No pipelining or other parallel processing techniques should be attempted. They would detract from the simple operation required for easy understanding, and would certainly conflict with goal #1.

Final: No change.

3.0 Instruction Set

3.1 Overview

There are 16 different P8 instructions. Research on instruction set usage was the basis for instruction selection. Each instruction has at least two addressing modes, with most of them having four. The instruction set is orthogonal, i.e., each instruction implements every relevant addressing mode.

3.2 Design Rationale

One of the requirements of a good instruction set is completeness. A complete instruction set can be used to evaluate any mathematical or logic function. The instruction set of an educational CPU should be complete and should illustrate the types of instructions normally supported by actual CPUs.

The instruction set should also be simple. There is research that indicates that even very simple instruction sets can be complete. As early as 1956 a simple 1-instruction computer was developed that met the requirement of completeness (Hayes, 1988, p. 210). Of course, an excessively simple instruction set requires very complicated programs to perform even the simplest tasks.

This instruction set is both complete and reasonably simple. Its completeness can be informally proven by writing a sequence of instructions to implement common operations that have not been included. For example, multiplication can be performed, even though there is no multiply instruction, with a sequence of adds and shifts. Similarly, the nonexistent AND function can be implemented because both OR and NOT are available.

Another desirable characteristic of an instruction set is regularity. A regular instruction set includes normally expected operations. In this case, some commonly encountered instructions have been omitted. This was done primarily for two reasons:

1. **The 74181 4-bit ALU used for this CPU has a limited repertoire of functions. For example, it implements a shift left and decrement, but not a shift right or increment. While this can lead to awkward code composition, and may sometimes require a little creative programming, it is possible to work around the missing functions.**

2. **This CPU is a teaching tool. A powerful instruction set is not a requirement. In fact, a simple instruction set, that implements only the most common instruction types is easier to understand. In the interest of simplicity, the instruction set has been limited to 16 different operations.**

Regularity also implies orthogonality. An orthogonal instructions set is one where each instruction can use every relevant addressing mode. This simplifies compiler design by making the rules for operand address specification consistent. While this is not an important consideration for the P8 CPU, it demonstrates the principle.

Research on instruction mixes for several types of processors has been compiled over the years. Of the various processors that have been benchmarked, the Intel 8086 most closely resembles the P8 CPU for this project (Hennessy & Patterson, 1990, p. 178). The frequency of instructions used by the 8086 is shown in Figure 3.1. This information is a reasonable guide for determining the most appropriate instruction mix for a simple, accumulator-based CPU like this one. Instructions that are used often should be included, if possible.

8086 Instruction	Average Frequency of Use
Move	27%
Conditional Jump	10%
Compare	7%
Push	7%
Pop	5%
Shift Left, Shift Right	5%
Loop	4%
Call	4%
Return	4%
Increment, Decrement	3%
Or, Xor	3%
Add	3%
Subtract	2%
Jump	2%

All other instructions had a frequency of use less than 2%.

Figure 3.1: Distribution of Instruction Frequencies on the Intel 8086.

Most instructions with a usage-frequency less than 2% have not been implemented. Additionally, there are several common instructions that cannot be fully implemented on the P8 CPU:

Conditional Jump:	The only condition flag is zero. Conditional jumps can only be based on whether the zero flag is set. This illustrates the principle while maintaining a simpler hardware implementation.
Compare:	Only equality can be evaluated.
Push & Pop:	Eliminating the stack pointer simplified the hardware significantly, but precluded the use of these instructions.

Call & Return:	**Again, the lack of a stack pointer makes these difficult to implement. The functions could be performed by jumping to predetermined addresses, but the programming would not be straightforward.**
Loop:	**This operation must be implemented with a sequence of simpler instructions.**
Increment:	**This function is not available on the 74181 ALU. The additional circuitry required to implement this would not contribute to the goal of simplicity. If an increment is required, it can be done by the add instruction with immediate data of 01h.**

Operations chosen for inclusion in this instruction set represent those used nearly two-thirds of the time in the 8086 benchmarks.

An educational CPU should also implement common addressing modes. Benchmark results of commonly used 8086 addressing modes for various application programs (Hennessy & Patterson, 1990, p. 177) can be seen in Figure 3.2.

8086 Addressing Mode	Frequency of Use
Memory	
Absolute	12%
Indirect	5%
Displacement	24%
Register	51%
Immediate	8%

Figure 3.2: Distribution of Addressing Mode Frequencies on the 8086.

Even though displacement addressing is very powerful, and used frequently on the 8086, it has not been implemented because hardware complexity would be increased.

3.3 Programmer's Model

The P8 CPU consists of three 8-bit registers and one 1-bit condition register. See Figure 3.3. It is an accumulator-based design. This means that the number in the accumulator (A register) is an operand for most ALU operations. Additionally, all ALU results are placed in the A register, and overwrite its previous contents. It is also used for reading input ports and writing output ports. The R register may be used to store general data or a second operand. It also contains the memory address of operands for instructions using indirect addressing.

Because the instruction pointer is an 8-bit register, only 256 bytes of memory may be addressed. Similarly, there are 256 available I/O ports, which do not share the memory space.

The Z register is a 1-bit register that stores the results of the last compare operation. If the result was zero, the Z register contains a "1", otherwise it is "0".

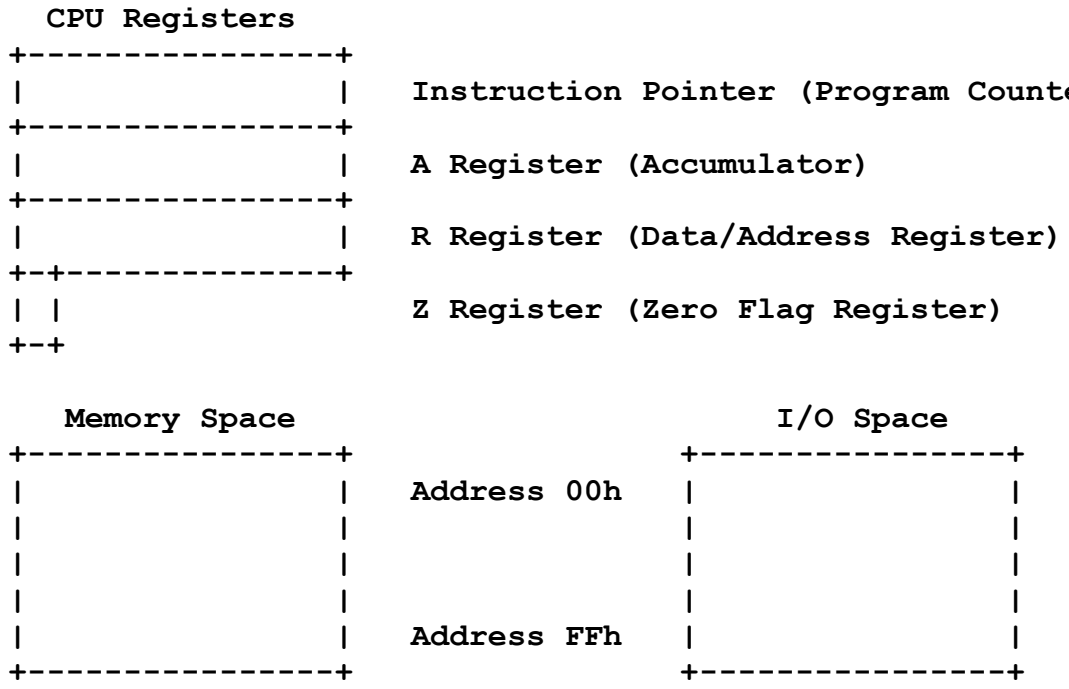


Figure 3.3: Programmer's Model of P8 CPU.

3.4 Instruction Types

P8 CPU instructions fall into five major instruction categories:

1. **Input / Output. These instructions transfer data between the accumulator and external I/O devices.**

   IN = Read Input Port
   OUT = Write Output Port

2. **Program Control. These instructions change the sequence of program execution. They are often called branch instructions.**

   JMP = Unconditional Jump
   JNZ = Jump If Not Zero (Conditional Jump)
   JZ = Jump If Zero (Conditional Jump)
   CMP = Compare (Sets / Resets Zero Bit For Conditional Jumps)

3. **Data Transfer. These instructions cause data in one location (either the internal registers or external memory) to be copied to another location.**

   LDA = Load A Register
   LDR = Load R Register
   STA = Store A Register
   STR = Store R Register

4. **Arithmetic. These instructions perform numerical operations on data. (Floating point operations are not supported.)**

   ADD = Add To A Register
   SUB = Subtract From A Register
   DEC = Decrement

5. **Logical. These instructions perform Boolean operations on data, including bit shifting.**

   OR = Or With A Register
   INV = Invert & Move To A Register
   SHL = Shift Left & Move To A Register

3.5 Addressing Modes

Four common addressing modes have been selected for this instruction set. They account for those used two-thirds of the time in Intel 8086 benchmarks. They are:

1. Direct. This is the same as absolute addressing. The address of the required data is part of the instruction. In this case, it will be the second byte of the instruction.

2. Indirect. The address containing the address of the required data is specified. There are normally two types of indirect modes: 1) memory-indirect; and 2) register-indirect. An instruction with memory-indirect addressing specifies the memory address in which the address of the required data is stored. A specified register contains the address of the data when register-indirect addressing is used. The indirect mode for this instruction set will be limited to register-indirect, and the register containing the address will always be the R Register.

3. Register. This is sometimes called inherent addressing. The required data is in a register.

4. Immediate. The required data is part of the instruction. For this architecture, it is the second byte of the instruction.

3.6 Instruction Format

Each instruction has an 8-bit opcode. The eight bits are divided into two fields: 1) the operation; and 2) the addressing mode (Figure 3.4).

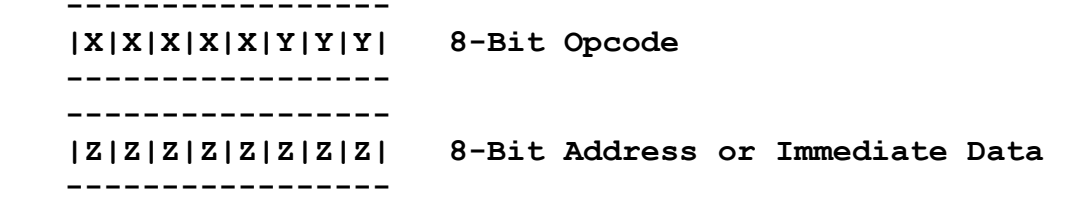


Figure 3.4: Instruction Format for 1-Byte & 2-Byte Instructions.

All instructions that use register addressing or indirect addressing require only one byte. A second byte is required for the other two addressing modes. The second byte of a direct instruction contains the 8-bit address, and the second byte of an immediate instruction specifies the immediate data.

The operation is encoded in the 5-bit field labeled 'XXXXX'.

5-Bit Operation Code	Instruction Type
00001	IN
00010	OUT
00100	JMP
00101	JNZ
00110	JZ
00111	CMP
01000	LDA
01001	LDR
01010	STA
01011	STR
01100	ADD
01101	SUB
01110	DEC
10000	OR
10001	INV
10010	SHL

The addressing mode is encoded in the 3-bit field labeled 'YYY'.

3-Bit Code	Addressing Mode	Data Location
000	Direct	Memory Address in Byte 2
001	Not Used	---
010	Register	A Register
011	Register	R Register
100	Indirect	Memory Address in R Register
101	Not Used	---
110	Immediate	Byte 2 of Instruction
111	Not Used	---

3.7 Instruction Set Repertoire

See Appendix 7.1 for a more comprehensive description of the instruction set.

Instruction	Addressing Mode	Operation Performed	Bytes
ADD address	Direct	A <- A + MEM(address)	2
ADD A	Register	A <- A + A	1
ADD R	Register	A <- A + R	1
ADD M	Indirect	A <- A + MEM(R)	1
ADD I, data	Immediate	A <- A + data	2
CMP address	Direct	If A - MEM(address) = 0: Z <- 1	2
CMP A	Register	If A - A = 0:Z <- 1	1
CMP R	Register	If A - R = 0: Z <- 1	1
CMP M	Indirect	If A - MEM(R) = 0: Z <- 1	1
CMP I, data	Immediate	If A - data = 0: Z <- 1	2
DEC address	Direct	A <- MEM(address) -1	2
DEC A	Register	A <- A - 1	1
DEC R	Register	A <- R - 1	1
DEC M	Indirect	A <- MEM(R) - 1	1
DEC I, data	Immediate	A <- data - 1	2
IN address	Direct	A <- PORT(address)	2
IN P	Indirect	A <- PORT(R)	1
INV address	Direct	A <- [MEM(address)]'	2
INV A	Register	A <- [A]'	1
INV R	Register	A <- [R]'	1
INV M	Indirect	A <- [MEM(R)]'	1
INV I, data	Immediate	A <- [data]'	2
JMP address	Direct	IP <- address	2
JMP R	Register	P <- R	1
JNZ address	Direct	If Z = 0: JMP address	2
JNZ R	Register	If Z = 0: JMP R	1
JZ address	Direct	If Z = 1: JMP address	2
JZ R	Register	If Z = 1: JMP R	1

Instruction	Addressing Mode	Operation Performed	Bytes
LDA address	Direct	A <- MEM(address)	2
LDA A	Register	A <- A	1
LDA R	Register	A <- R	1
LDA M	Indirect	A <- MEM(R)	1
LDA I, data	Immediate	A <- data	2
LDR address	Direct	R <- MEM(address)	2
LDR A	Register	R <- A	1
LDR R	Register	R <- R	1
LDR M	Indirect	R <- MEM(R)	1
LDR I, data	Immediate	R <- data	2
OR address	Direct	A <- A C MEM(address)	2
OR A	Register	A <- A C A	1
OR R	Register	A <- A C R	1
OR M	Indirect	A <- A C MEM(R)	1
OR I, data	Immediate	A <- A C data	2
OUT address	Direct	PORT(address) <- A	2
OUT P	Indirect	PORT(R) <- A	1
SHL address	Direct	A <- $[MEM(address)]_{6..0}$ ## 02	
SHL A	Register	A <- $[A]_{6..0}$ ## 0	1
SHL R	Register	A <- $[R]_{6..0}$ ## 0	1
SHL M	Indirect	A <- $[MEM(R)]_{6..0}$ ## 0	1
SHL I, data	Immediate	A <- $[data]_{6..0}$ ## 0	2
STA address	Direct	MEM(address) <- A	2
STA M	Indirect	MEM(R) <- A	1
STR address	Direct	MEM(address) <- R	2
STR M	Indirect	MEM(R) <- R	1
SUB address	Direct	A <- A - MEM(address)	2
SUB A	Register	A <- A - A	1
SUB R	Register	A <- A - R	1
SUB M	Indirect	A <- A - MEM(R)	1
SUB I, data	Immediate	A <- A - data	2

4.0 Hardware

4.1 Overview

A CPU consists of a data section (often called a datapath) and a control section. The P8 CPU contains a simplified version of each.

The datapath contains registers, an ALU, and an internal system bus. Registers are very fast memory locations that are internal to the CPU and separate from the main system memory. They can be used as "scratch pads" during calculations. The ALU is a combinational logic device that performs arithmetic and logical operations on data. The internal bus permits data exchange between the ALU and registers.

The control section interprets each instruction to be executed and asserts the datapath's control signals in the proper sequence to implement the instruction. After executing an instruction, it proceeds to the next instruction.

4.2 Design Rationale

The following requirements and constraints were placed upon the hardware. In some cases they were contradictory, and a compromise was reached that favored reduced cost and/or construction time.

- It must, of course, meet the specifications outlined in Section 2.5 and implement the complete instruction set described in Section 3.7. No compromise on this point is permissible.
- Use only standard logic and memory devices. This includes programmable devices, such as PALs and PROMs.
- To save space and wiring time, use the fewest number of IC devices that is practical.
- To reduce costs, use parts that are already on-hand whenever possible.
- The operation of the datapath section and control section should be simple to understand, i.e., no "clever" designs that are difficult to grasp.

- **The CPU circuitry should be separate from supporting circuitry, such as memory, I/O ports, oscillators, etc.**

4.3 Programmer's Model

The programmer's model defines the architecture of a computer. It is a high-level "machine" that is visible to the programmer, but independent of the actual physical hardware. It deals with the functional behavior of the computer as seen by the programmer. It includes the number and sizes of registers, types of instructions, addressing modes, and available memory. See Section 3 for more information on many of the architectural features of the P8 CPU.

The P8 CPU consists of three 8-bit registers and one 1-bit condition register. See Figure 4.1. It is an accumulator-based design. This means that the number in the accumulator (A register) is an operand for most ALU operations. Additionally, all ALU results are placed in the A register, and overwrite its previous contents. It is also used for reading input ports and writing output ports. The R register may be used to store general data or a second operand. It also contains the memory address of operands for instructions using indirect addressing.

Because the instruction pointer is an 8-bit register, only 256 bytes of memory may be addressed. Similarly, there are 256 available I/O ports, which do not share the memory space.

The Z register is a 1-bit register that stores the results of the last compare operation. If the result was zero, the Z register contains a "1", otherwise it is "0".

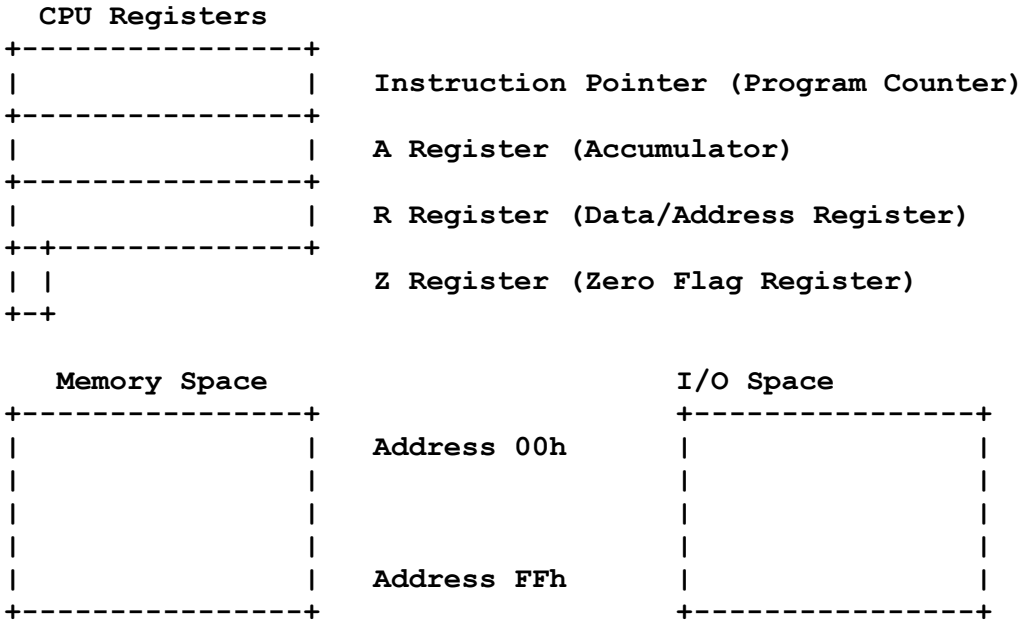


Figure 4.1: Programmer's Model of P8 CPU.

4.4 Functional Block Model

Beneath the high-level programmer's model, are the functional units that implement it. These are the control unit, internal system busses, ALU, and registers (many of which cannot be directly manipulated users). Figure 4.2 illustrates the purposes of the CPU's functional units and the relationships between them. These functional units implement the instruction cycle, which fetches an instruction from external memory, decodes it, reads the required operand(s), and then executes the instruction. Most instructions require several steps (microinstructions) to be performed by the CPU.

The control section consists of the zero condition register (Z) and a block labeled "control logic". The details of the control logic's actual implementation is unimportant at this level. The diagram clearly shows, however, that it has inputs from the instruction register (IR) and Z. These inputs specify the outputs generated by the control logic. IR sends eight bits to the control unit that identify the instruction to be executed. If it is a conditional instruction, i.e., it depends on the logic level of Z, the condition bit will combine with the inputs from IR to determine the proper sequence of control signals.

There are two types of control logic outputs. One type is for internal use. This type consists of the datapath control bits (C_0 - C_n). These control bits load registers, specify the type of ALU operation to be performed, and allow data on the internal system busses at the proper time. The other type is used by external circuitry, such as memory and I/O ports. These bits (MEMR, MEMW, IOR, IOW) control the data flow on external system busses between the CPU and its peripheral devices.

The 8-bit internal bus (IB) is the primary path for data flow between the functional units. Nearly every functional unit is connected either directly, or indirectly, to IB. Using one internal system bus results in a simple hardware design that is ideal for an educational CPU. Some secondary internal busses are also present, but their connections are restricted to a small number of functional units.

The ALU can perform the following operations:

- Compare
- Add
- Subtract
- Decrement
- Logical OR
- Logical NOT (Invert)
- Logical Shift Left

The results of most ALU operations are placed in the A register, which is the accumulator. The sole exception to this is the compare operation, which is intended only to set/reset Z. Even if an ALU result is ultimately destined for another location, it must first pass through the accumulator. The output of the accumulator is also permanently connected to one of the ALU 8-bit inputs. This means that the value in the accumulator is always an operand for ALU functions that require two operands. It is also the source for many single-operand functions. This feature allows a very simple hardware design. In fact, it is used by many commercial 8-bit microprocessors and single-chip microcontrollers for that very reason. It is, however, unsuitable for high-end CPUs because it is inefficient.

The R register can be used for general data storage. Its main purpose in this CPU, however, is to illustrate two addressing modes that require an additional register. The first is register

addressing, where the required data are in a register. The second is indirect addressing, or in this specific case, register-indirect addressing. In this mode, the R register contains the address where the data are located. This addressing mode is implemented by transferring the contents of the R register to the operand register (OR), from which the address register (AR) is loaded.

The buffer connected to the output of the A register allows the accumulator's output to always be available to the ALU without interfering with IB. The buffer connected to the R register is actually not required for proper operation, but it was included to allow constant monitoring of the R register's contents. This was important because of the educational nature of the CPU.

The CPU accesses the external data bus with the data register (DR). DR, like the external data bus, is bi-directional. Data to be placed on the external data bus are sent to DR from either the A register or the R register along IB. Data placed in DR by the external data bus can be read by the following functional units:

- Operand Register
- Instruction Register
- ALU
- A Register (through ALU)
- R Register

The external address bus is accessed through the address register (AR). The 8-bit address in AR is placed on the external address bus. An address can be written to AR by either the instruction pointer (IP) or the operand register (OR). IP is an 8-bit binary counter that keeps track of the memory address containing the next instruction. IP increments after each instruction is read, and therefore contains the address of the instruction in the next sequential memory location. IP can also be preset to any 8-bit address placed on IB when a branch instruction is executed. OR is loaded with the memory address, or I/O port address, of required data when an instruction using direct addressing, or indirect addressing, is executed.

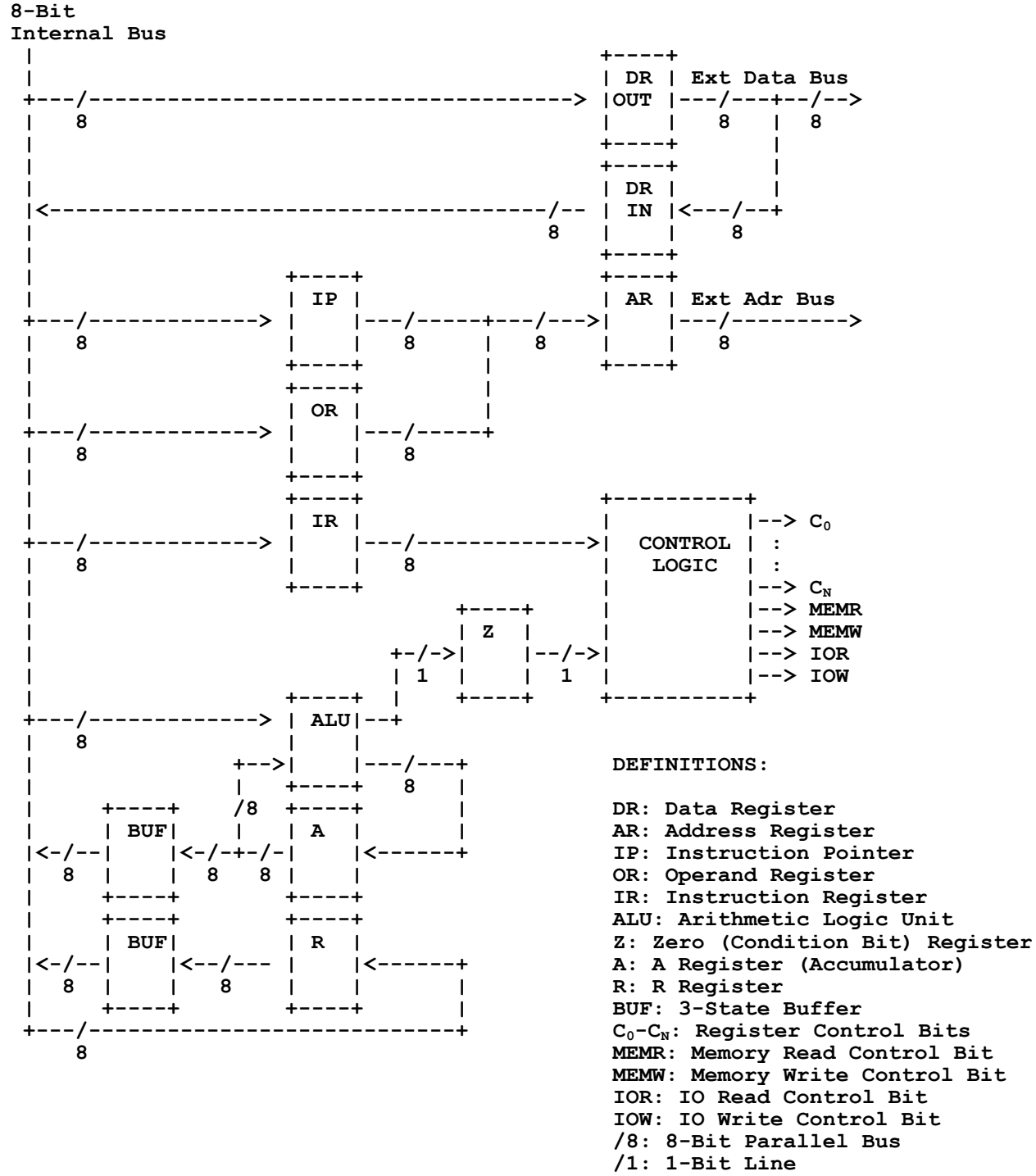


Figure 4.2: Functional Block Diagram of P8 CPU

4.5 Detailed Design

This section discusses the specific design details of each CPU functional block. See the Appendix for a complete set of schematic diagrams and assembly drawings, as well as the code listings for the programmable devices used in the P8 CPU.

4.5.1 Datapath

The datapath contains four user-visible registers (A, R, IP, and Z), four registers that are not visible to users (DR, AR, OR, and IR), an ALU, and the connecting busses between them. The actual implementation of the datapath will be explained here.

Register and Buffer Selection

Seven 8-bit registers, and two 8-bit tri-state buffers are required for the datapath. It actually takes 11 ICs to implement them. The requirements of each register, and the TTL device(s) used to implement each one is listed below:

Operand Register - OR holds the address of an operand that must be read from memory.

Requirements: 8 bits
edge-triggered load
tri-state output

Suitable TTL Device: 74LS374

Implementation Note: See Figure 4.3.

Address Register - AR writes port or memory addresses to the external address bus.

Requirements: 8 bits
edge-triggered load
latched tri-state output

Suitable TTL Device: 74LS374

Implementation Notes: Latch output enable control to achieve latched tri-state output. See Figure 4.3.

Data Register - DR is a bi-directional register that writes data to the external data bus. It also reads data from the external data bus.

Requirements:
8 bits
bi-directional
edge-triggered load
tri-state output to internal bus
latched tri-state output to external bus

Suitable TTL Device: 74LS374 (2 required)

Implementation Notes: Use 2 uni-directional registers, DR(IN) and DR(OUT). Latch output enable control to achieve latched tri-state output. See Figure 4.3.

Instruction Pointer - IP keeps track of the next instruction to be read from memory.

Requirements:
8 bits
edge-triggered load
edge-triggered increment
clear
tri-state output

Suitable TTL Device: 74LS163 (2 required)

Implementation Notes: Use 2 4-bit binary counters. These counters meet all of the requirements except for the tri-state outputs. Use a single 74LS244 to buffer the eight IP output bits. See Figure 4.3.

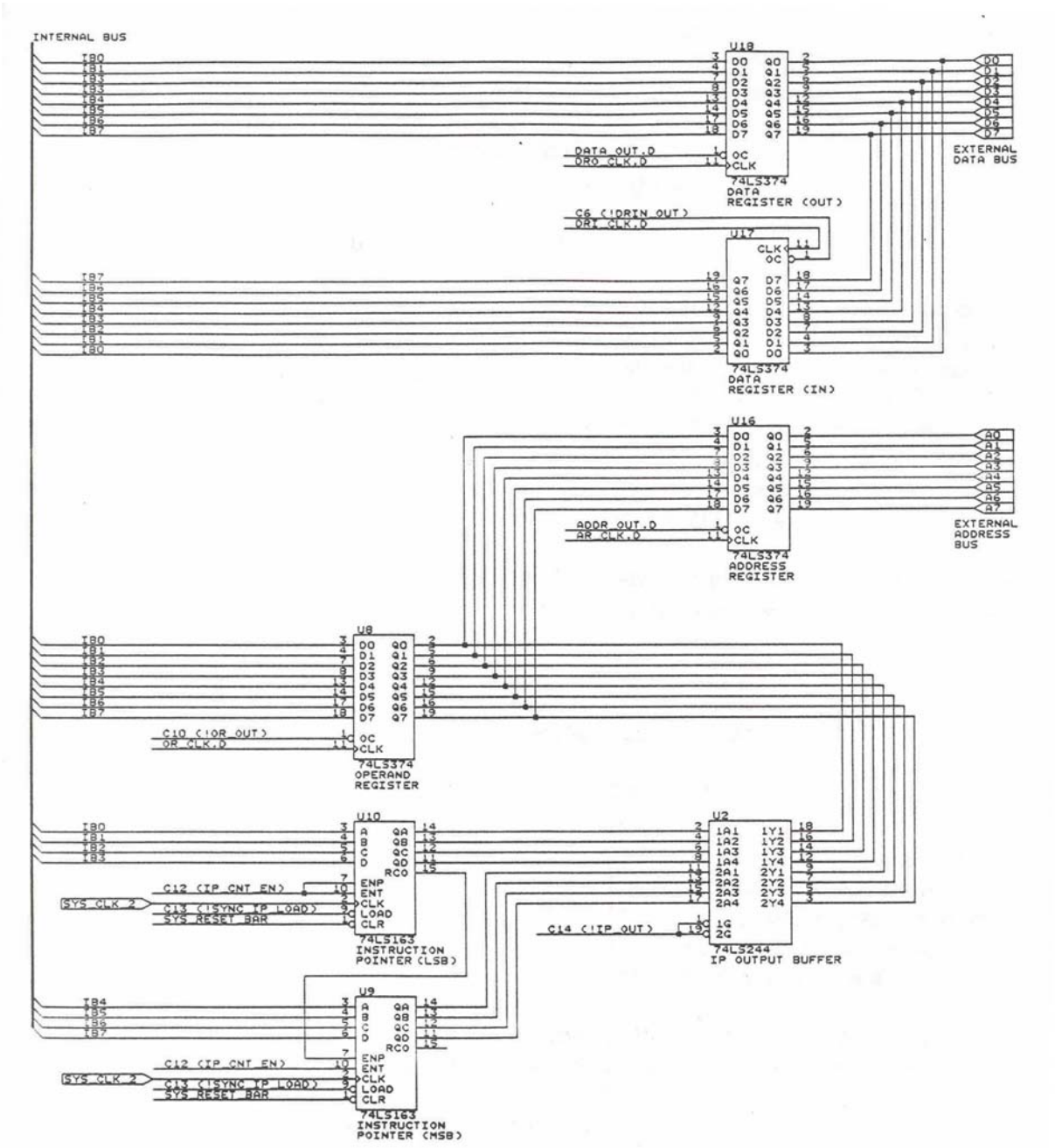


Figure 4.3: Schematic of Datapath with Operand Register, Address Register, Data Register, and Instruction Pointer

A Register - The A register is the accumulator. It is the destination for all ALU results. It is also a source for all 2-operand instructions, and many 1-operand instructions.

Requirements:	8 bits edge-triggered load
Suitable TTL Device:	74LS374
Implementation Notes:	Permanently enable the tri-state outputs. A 74LS273 could also have been used. The A register is isolated from IB with a 74LS244 (8-bit, tri-state buffer). This allows the outputs of the A register to be continuous inputs to the ALU. See Figure 4.4.

R Register - The R register can be used to store data, hold an ALU operand, or point an operand when using the indirect addressing mode.

Requirements:	8 bits edge-triggered load
Suitable TTL Device:	74LS374
Implementation Notes:	See "Implementation Notes" for the A register. The separate buffer (74LS244) is only required to monitor the contents of the R register. See Figure 4.4.

Instruction Register - IR contains the op code of the current instruction. Its outputs are connected to the control store address bits.

Requirements:	8 bits edge-triggered load synchronous clear
Suitable TTL Device:	74LS273
Implementation Note:	See Figure 4.4.

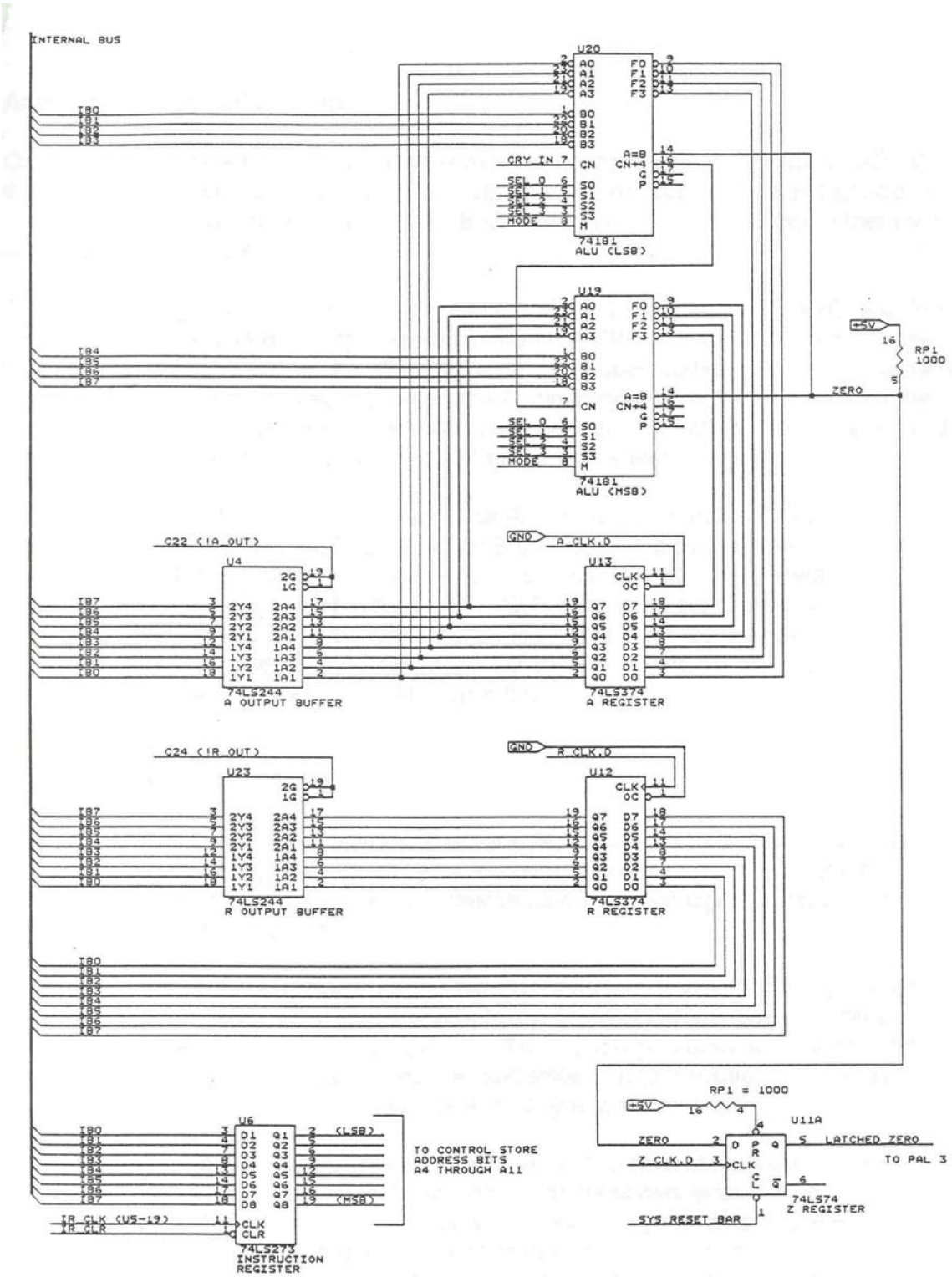


Figure 4.4: Schematic Diagram of Datapath with A Register, R Register, Instruction Register, Z Register, and ALU

Arithmetic Logic Unit Selection

One 8-bit ALU is needed to implement the arithmetic and logical instructions. Actually, the decision of which ALU instructions to include in the instruction set was dictated by the only TTL ALU available -- the 74181. Because this is a 4-bit ALU, two of them are required. See Figure 4.4.

This ALU can generate condition bits for carry, half-carry, A=B (or zero), A>B, and A<B. Only the zero condition bit is implemented in the MP8 CPU. This is sufficient to show how conditional branching works. The "A=B" output is open collector, and must be tied to an external resistor. It goes high when the A and B inputs are the same during the execution of an P8 compare instruction. (The compare instruction is actually an ALU subtract operation, but the results are not placed in the A Register.)

The "C_N+4" output of the least significant 4-bit ALU must be connected to the "C_N" input of the most significant 4-bit ALU to create an 8-bit ALU. This connection allows a carry between bit 3 (MSB of first ALU) and bit 4 (LSB of second ALU) and generates the correct 8-bit result during an ALU operation. "C_N" of the least significant ALU must be low for a subtract operation and high for all others. This input, as well as the select and mode inputs, are asserted by the control logic of the CPU. The control logic properly configures all inputs for the desired ALU operation.

4.5.2 Control Logic

The control logic accepts inputs from IR and the ALU (through Z) and generates as output all the control signals necessary for the correct operation of the datapath. There are two approaches that can be taken in designing the control logic: 1) hardwired control, and 2) microcoded control.

A hardwired controller is a state machine with flip flops to maintain state information and combinational logic at the inputs and outputs. Control outputs are generated for each state by the combinational output logic. The input logic combines the input bits with the present state bits to determine the next state. State transitions, and their corresponding control outputs, are synchronized by a system clock.

A microcode controller consists of a microprogram in a control store memory (usually ROM) that is executed by a simple

microprogram address sequencer. The microprogram is composed of microwords, which are simply the data bits in each memory location. Each microword can perform one or more microinstructions, which comprise each of the P8 instructions. As each microprogram step is addressed, the data bits of the microword for that step are asserted. They are connected to a pipeline register (flip flops) where they are synchronized with the system clock. The synchronized outputs of the pipeline register are used to control the datapath and are called control bits.

Each type of controller has advantages and disadvantages (Heuring & Jordan, 1997, p. 244):

- Speed: Hardwired control is much faster. It has a latency of just a few gate delays, considerably less than microcoded control units, which must perform a memory fetch for each control step.

- Ease of prototyping: The microcode design approach wins here, since it is generally easier to reprogram a memory chip than to rewire logic.

- Flexibility of use: Once again, microcoding is superior when the designer wishes to change instruction sets. This might be desirable for instruction set upgrades or when several instruction sets are to be emulated.

Microprogrammed control was selected for this CPU because it is simpler to implement, and more easily modified than hardwired control. It is slower than hardwired control, but that is not a serious disadvantage for the P8 CPU.

Control Store Configuration

Thirty-one control signals are required to operate the P8 CPU. Because industry-standard PROMs are 8 bits wide, four of them are used to create the control store with a 32-bit microword. This leaves one control bit (C_{29}) unused. The control store PROMS need 12 address bits. Eight are used by the opcode of the instruction to be executed, three are needed to address the individual microwords (up to 8) that comprise each instruction, and one bit is used to determine the action of conditional instructions. The 2732 (4K X 8) PROM would have been ideal for this application, but none were available. The 2764 (8K X 8) PROM was substituted. The

most significant address bits of the 2764 PROMs are disabled by grounding them. These PROMs are not used very efficiently for this application, but at the board level, the amount of wasted silicon area is not important.

Each P8 instruction is assigned a block of 16 consecutive addresses within the control store. See Figure 4.5 for a block diagram of how the control store is addressed. Each instruction's block is selected with its 8-bit opcode, which is connected to the eight most significant control store address bits. Address bit 3 is connected via some enabling logic in PAL 3 to the condition bit register (Z). This divides the instruction's control store block into two sub-blocks of eight addresses each. Most instructions use only the first sub-block ($A_3 = 0$). Conditional instructions, however, have microwords coded into both sub-blocks. The value of the condition bit, and, therefore, A_3, determines which set of microwords will be executed. This allows the same instruction to execute differently for each value in Z. The three least significant control store address bits are connected to a binary counter called the microinstruction pointer (MIP). This counter sequences through each of the microwords that contain microinstructions required to execute the instruction in IR.

Figure 4.6 shows the control store code that implements the "Jump If Not Zero" Instruction. The 5-bit operation and 3-bit addressing mode fields of the 8-bit opcode (28h) are combined with the condition bit (Z) and the 3-bit MIP output to form the 12-bit control store address. If the condition bit is not set (Z = 0) the jump is executed. It takes three microwords, which encode a total of eight microinstructions, to complete. If Z = 1, the jump is not executed, and the only microinstruction necessary is to reset IR.

The purpose of each of the 31 control bits is clearly labeled in Figure 4.6. They can be traced to the various registers in the schematic diagram.

The following microinstructions are executed if Z = 0:

Step	Microinstruction	Control Bit(s) Asserted
0	AR <-- IP	C_{27},* C_{14}, C_5
	Memory Address <-- AR	C_4
	Assert MEMR	C_2
1	DR <-- Memory Data	C_{27},* C_7
	Memory Address <-- AR	C_4
	Assert MEMR	C_2
2	IP <-- DR	C_{27},* C_{13}, C_6
	Reset IR (For FETCH)	C_0

*** C_{27} is set during all conditional instructions to enable A_3. Notice that the first two microwords each contain three microinstructions. The third microword contains two microinstructions, for a total of eight. A new microword is asserted on each cycle of the system clock.**

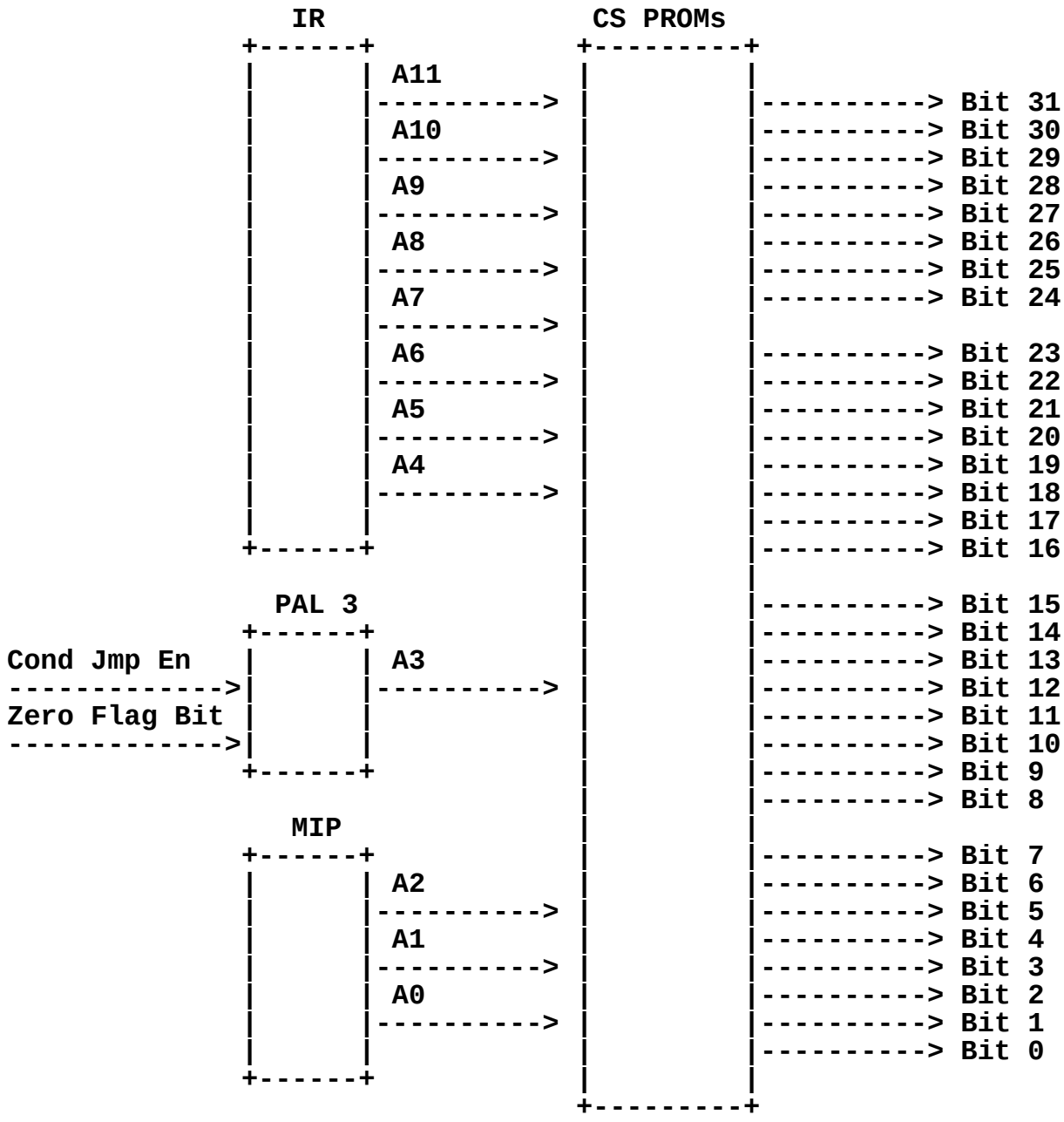


Figure 4.5: Block Diagram of Control Store Addressing

JNZ Address

Address				Data																															
				IOW	IOR	SPARE	COMPARE	COND JMP EN	ZLOAD	RLOAD	!ROUT	ALOAD	!AOUT	SHL	INV	OR	DEC	SUB	ADD	PASS THRU	!IPOUT	!SYNC IPLOAD	IPCLK EEN	ORLOAD	!OROUT	DROUTLOAD	!DROUTOUT	DRINLOAD	!DRINOUT	ARLOAD	!AROUT	MEMW	MEMR	ILOAD	IRRESET
OP	ADR	Z	MIP	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
00101	000	0	000	0	0	0	0	1	0	0	1	0	1	0	0	0	0	0	0	0	0	1	0	0	1	0	1	0	1	1	0	0	1	0	0
0			001	0	0	0	0	1	0	0	1	0	1	0	0	0	0	0	0	0	1	1	0	0	1	0	1	1	1	0	0	0	1	0	0
0			010	0	0	0	0	1	0	0	1	0	1	0	0	0	0	0	0	0	1	0	1	0	1	0	1	0	0	0	1	0	0	0	1
0			011																																
0			100																																
0			101																																
0			110																																
0			111																																
1			000	0	0	0	0	1	0	0	1	0	1	0	0	0	0	0	0	0	1	1	1	0	1	0	1	0	1	0	1	0	0	0	1
1			001																																
1			010																																
1			011																																
1			100																																
1			101																																
1			110																																
1			111																																

Figure 4.6: Control Store Code For "Jump If Not Zero"

For a complete listing of microinstructions and control store code, see Appendix 7.1 and Appendix 7.2.

The schematic diagram of the control store is shown in Figure 4.7.

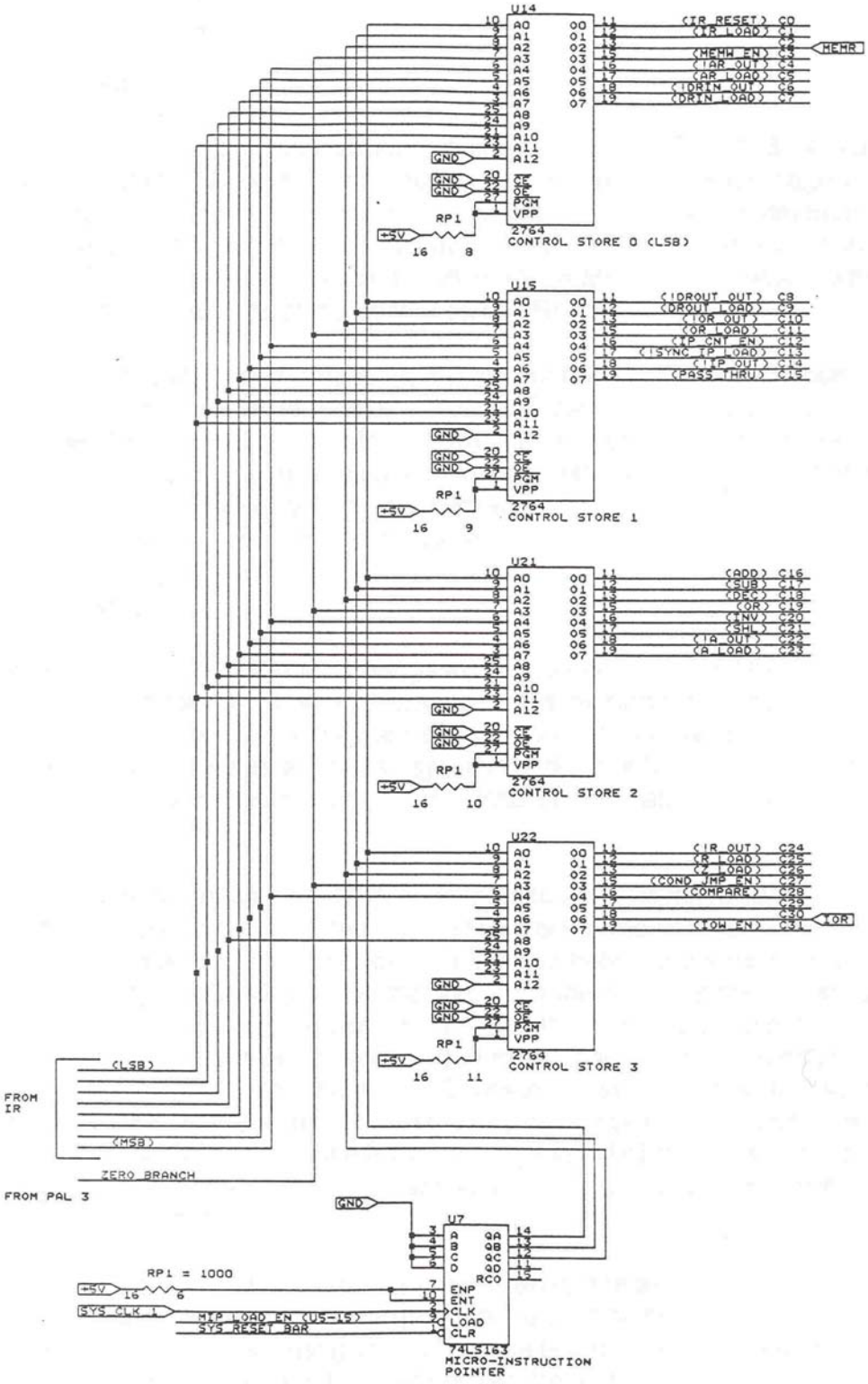


Figure 4.7: Schematic Diagram of Control Store

Control Store Sequencer

The control store sequencer, or microinstruction pointer (MIP), is a 74LS163. It is a 4-bit counter with a synchronous load, synchronous reset, and positive edge triggering. Its purpose is to step through the consecutive microwords for each P8 instruction that executes. Only three of its four bits are used, because no instruction has more than eight microwords. The three least significant outputs of the MIP are connected to the three least significant address bits of the control store PROMs.

The count enables of the MIP are permanently set to count on each positive clock edge. When clocked while the system reset is asserted, the count operation is overridden and the MIP is reset to zero. This allows it to address the first microword of the FETCH operation after the reset is released. On the last microinstruction of FETCH or any P8 instruction, the load enable is asserted, which overrides the count enables and loads the MIP with zero on the next clock pulse.

Control Signal Timing

When the control store's address inputs change in order to execute a different microword, all of the control store's data bits change. It is important that these bits settle to the correct logic level before they are allowed to affect the datapath. To ensure this happens, the P8 has a 2-phase system clock. The MIP is connected to phase 1 (CLK1). The pipeline register (PAL 1 and PAL 3), and datapath are connected to phase 2 (CLK2).

When CLK1 goes from low to high, the MIP increments to the next microword. The control bits of that microword are asserted, but no registers in the datapath change states until the rising edge of CLK2. Enough time elapses between the rising edge of CLK1 and the rising edge of CLK2 to allow the outputs of the control store to stabilize before they are acted upon. This is necessary because the control bits do not all assert simultaneously. Most of them are connected to the inputs of positive edge triggered flip flops in PAL 1 and PAL 3 until CLK2, which is also connected to PAL 1 and PAL 3, clocks. When CLK2 asserts, all of the register-clocking control bits are allowed through the pipeline register simultaneously so that all of the registers in the datapath change states together. Figure 4.8 shows PAL 1 and PAL 3, along with the logic equation of each output.

Not all control bits must wait for CLK2. Some of them enable registers that are clocked directly by CLK2. These registers do not respond to the control inputs until the rising edge of CLK2, so the control bits must go directly to the registers and be stable before CLK2 arrives. Other control bits do not operate on registers at all. The control bits that determine the operation to be performed by the ALU are examples. The ALU control bits go to PAL 2, which is a custom encoder that asserts the correct select and mode bits of the ALU. PAL 2 is also shown in Figure 4.8.

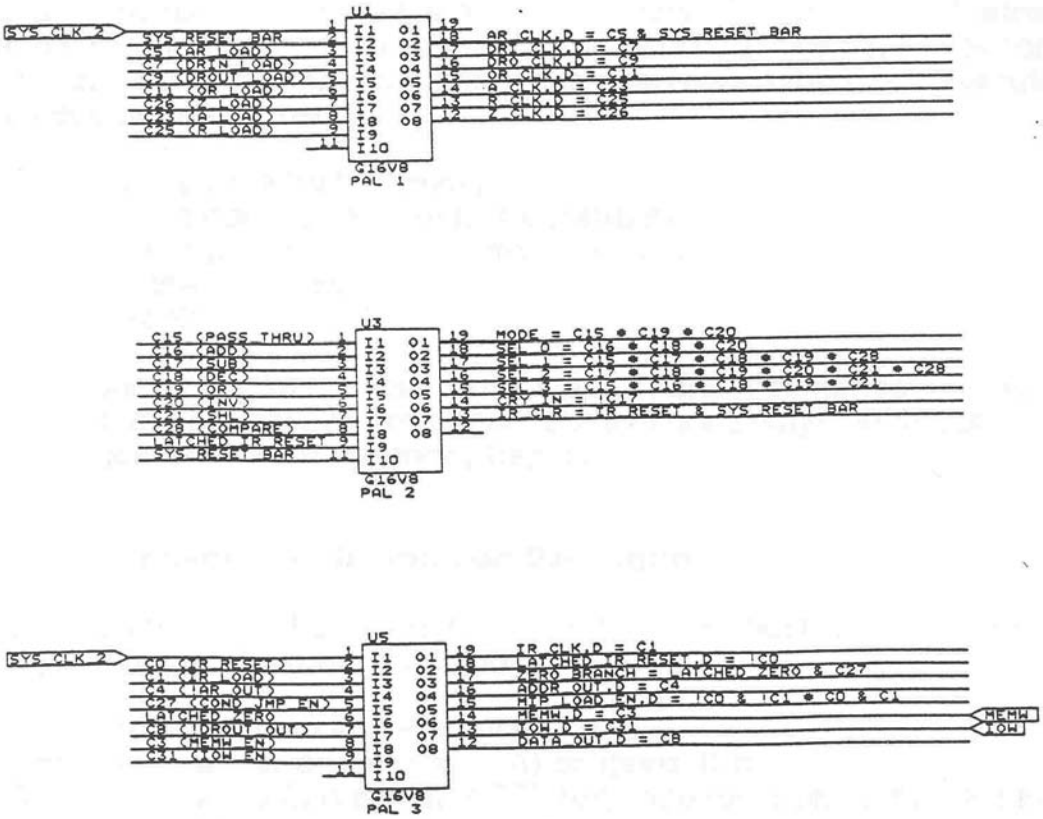


Figure 4.8: PAL 1, PAL 2 and PAL 3

After a system reset, the MIP contains 0h and IR contains 00h. This will execute the first microinstruction of FETCH during the next CLK2. To ensure that CLK2 is the first clock pulse after a system reset, a D flip flop is used as a reset synchronizer. See Figure 4.9.

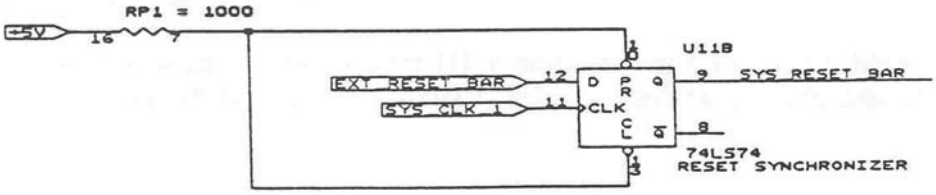


Figure 4.9: Reset Synchronizer

5.0 Operation

5.1 The Instruction Cycle

The CPU executes instructions that are stored in memory. The process of retrieving instructions and executing them is called the instruction cycle. Because it is comprised of a fetch operation and an execute operation, it is often called the fetch-execute cycle.
The following steps are performed:

1. Fetch an opcode from memory.
2. Decode the opcode to identify the instruction.
3. Read the required operand(s) from memory.
4. Execute the instruction.
5. Return to Step 1.

Steps 1 and 2 are part of the fetch cycle, while steps 3, and 4 comprise the execution cycle. Steps 1, 2, and 4 are always performed, but it is not always necessary to retrieve additional operands from memory (Step 3).

5.2 Step-By-Step Example of Instruction Execution

Examining the complete instruction cycle for one instruction illustrates the operation of the P8 CPU. Assume the following conditions:

- The instruction pointer (IP) contains 07h
- The A register, or accumulator (A) contains 13h
- Memory address 07h contains 68h (op code for "Subtract Direct From A")
- Memory address 08h contains 1Ah
- Memory address 1Ah contains 11h

The P8 instruction SUB 1Ah means "Subtract the data in memory location 1Ah from the A register and place the results in the A register". It has the opcode 68h. The second byte of the instruction (1Ah in this case) is the address where the operand will be found. The following pages describe the execution of this instruction.

Fetch Cycle

The first part of the instruction cycle fetches the opcode from memory, places it in the instruction register, and decodes it.

At the beginning of the fetch operation IR contains a 00h and MIP holds a 0h. This sets the control store's address to 000h. The CPU is ready to proceed with Step 1 of the fetch operation.

Step 1

The control bits in CS address 000h trigger the following datapath events:

Event	Control Bits
AR <-- IP	!IP_OUT, AR_LOAD
External Address Bus <-- AR	!AR_OUT
Assert MEMR	MEMR

The output of IP is enabled and the 07h in IP is placed on the inputs of AR. On CLK2, 07h is loaded into AR. The output of AR is enabled and 07h is placed on the external address bus. MEMR is asserted. See Figure 5.1. MIP increments the CS address to 001h on CLK1.

Step 2

The control bits in CS address 001h trigger the following datapath events:

Event	Control Bits
External Address Bus <-- AR	!AR_OUT
Assert MEMR	MEMR
DR <-- Op Code	DR(IN)_LOAD
IP <-- IP + 1	IP_CLK_EN

The output of AR continues to be enabled and 07h remains on the external address bus. MEMR remains asserted. On CLK2, the 68h, which was in memory location 07h, is loaded into DR. IP is incremented to 08h. See Figure 5.2. MIP increments the CS address to 002h on CLK1.

Step 3

The control bits in CS address 002h trigger the following datapath events:

Event	Control Bits
IR <--DR	!DR(IN)_OUT, IR_LOAD

The output of DR is enabled and 68h is placed on the internal bus (IB). On CLK2, the 68h is loaded into IR, and "0" is placed on the active-low LOAD input of MIP. See Figure 5.3. MIP loads 0h and the CS address goes to 680h on CLK1. The instruction is "decoded" when the op code in IR is placed on the address bus of CS.

Fetch Cycle Step 1

(AR <-- IP; External Address Bus <-- AR; Assert MEMR)

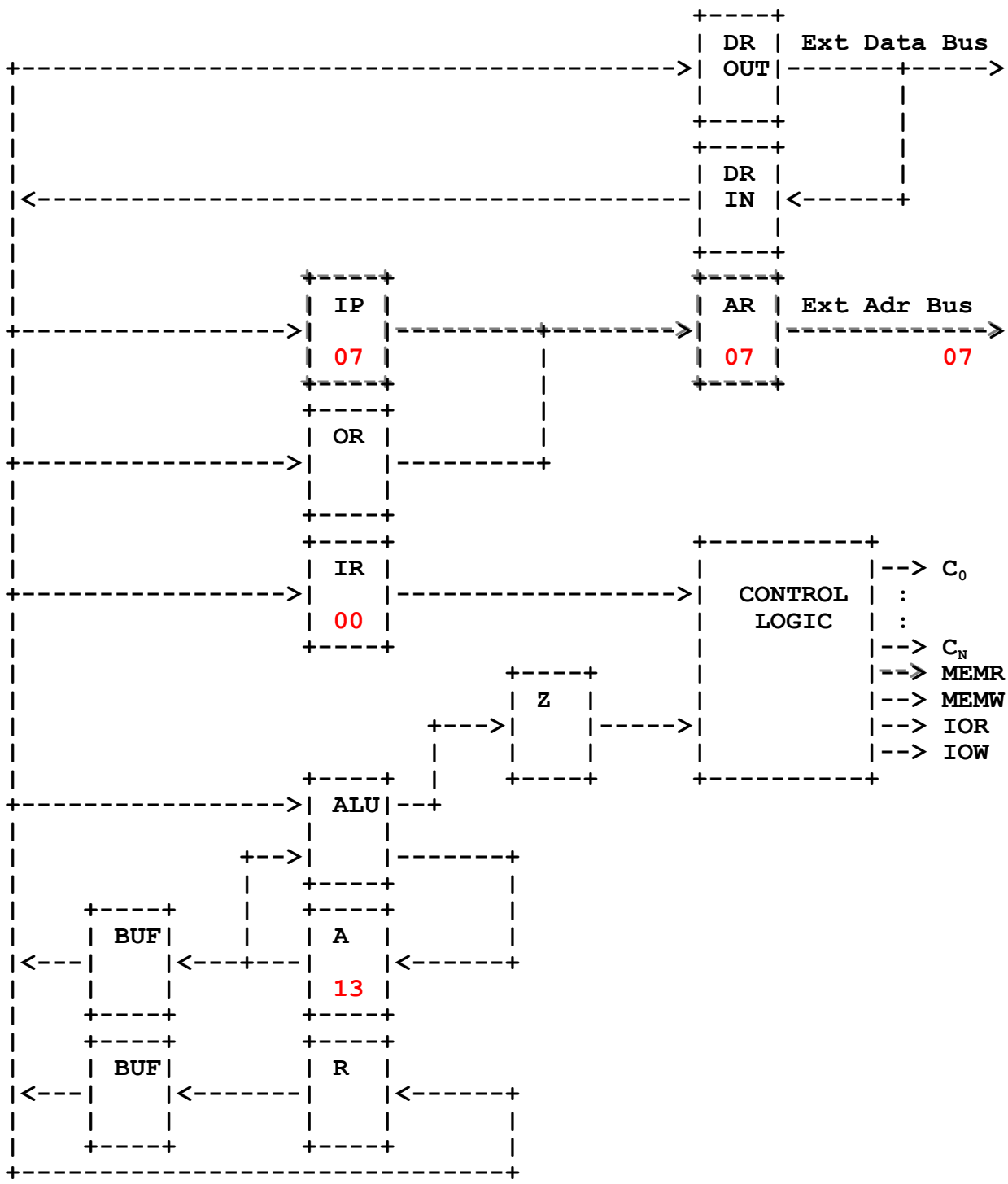


Figure 5.1: Instruction Cycle (Fetch 1) for SUB 1Ah

Fetch Cycle Step 2

(External Address Bus <-- AR; Assert MEMR; DR <-- Op Code;
IP <--IP + 1)

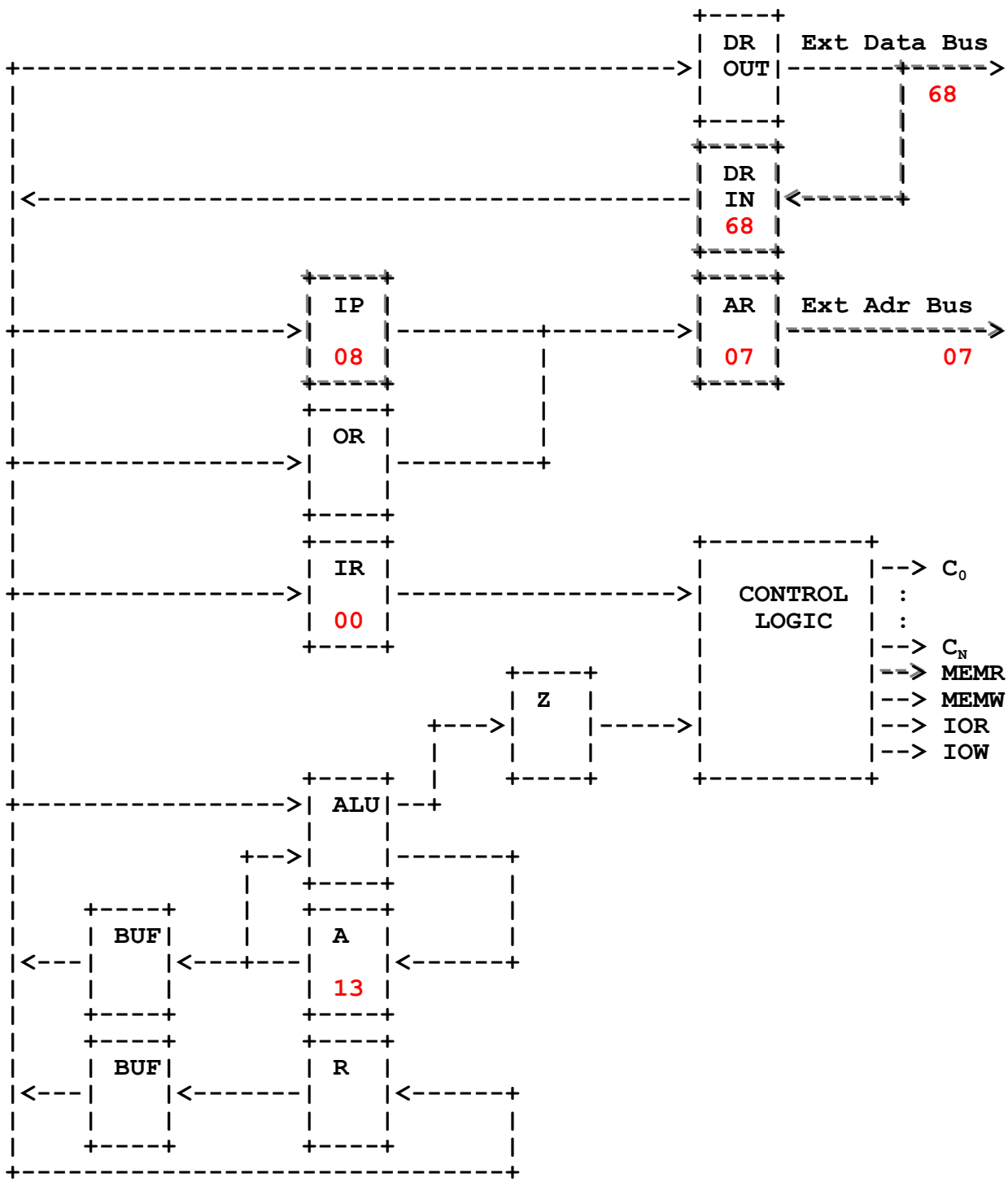


Figure 5.2: Instruction Cycle (Fetch 2) for SUB 1Ah

Fetch Cycle Step 3

(IR <-- DR)

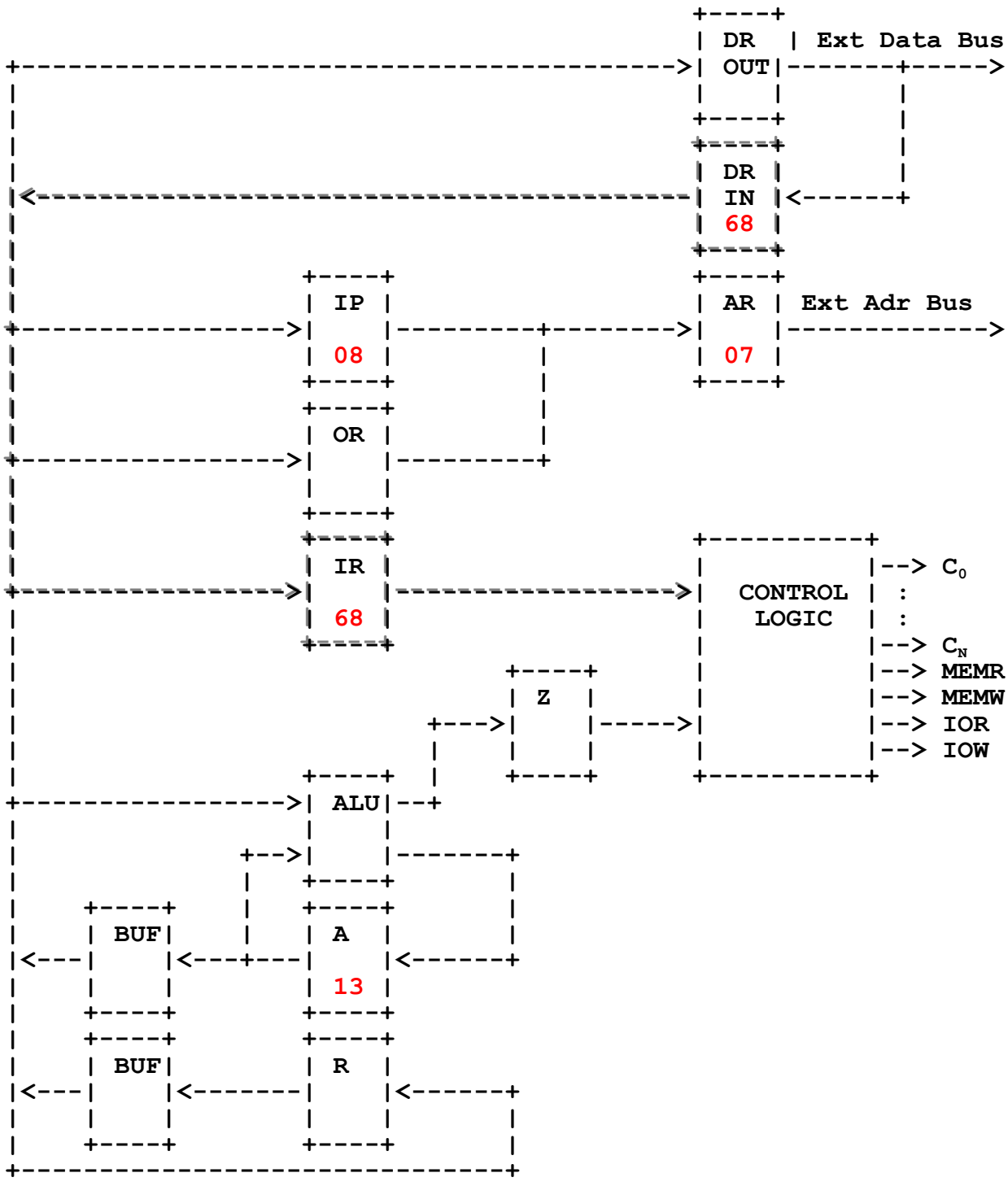


Figure 5.3: Instruction Cycle (Fetch 3) for SUB 1Ah

Execute Cycle

The remaining steps of the instruction cycle retrieve the operand from memory, execute the ALU operation, and store the result back in the accumulator. In this instruction, the next byte after the op code contains the memory address where the data are located. Two memory accesses are, therefore, required -- once to get the address, and a second time to get the data.

Step 1

The control bits in CS address 680h trigger the following datapath events:

Event	Control Bits
AR <-- IP	!IP_OUT, AR_LOAD
External Address Bus <-- AR	!AR_OUT
Assert MEMR	MEMR

The output of IP is enabled and the 08h in IP is placed on the inputs of AR. On CLK2, 08h is loaded into AR. The output of AR is enabled and 08h is placed on the external address bus. MEMR is asserted. See Figure 5.4. MIP increments the CS address to 681h on CLK1.

Step 2

The control bits in CS address 681h trigger the following datapath events:

Event	Control Bits
External Address Bus <-- AR	!AR_OUT
Assert MEMR	MEMR
DR <-- Byte # 2 (Address of Data)	DR(IN)_LOAD
IP <-- IP + 1	IP_CLK_EN

The output of AR continues to be enabled and 08h remains on the external address bus. MEMR remains asserted. On CLK2, the 1Ah, which was in memory location 08h, is loaded into DR. IP is incremented to 09h. See Figure 5.5. MIP increments the CS address to 682h on CLK1.

Step 3

The control bits in CS address 682h trigger the following datapath events:

Event	Control Bits
OR <--DR	!DR(IN)_OUT, OR_LOAD

The output of DR is enabled and 1Ah is placed on the internal bus (IB). On CLK2, the 1Ah is loaded into the operand register (OR). See Figure 5.6. MIP increments the CS address to 683h on CLK1.

Execute Cycle Step 1

(AR <-- IP; External Address Bus <-- AR; Assert MEMR)

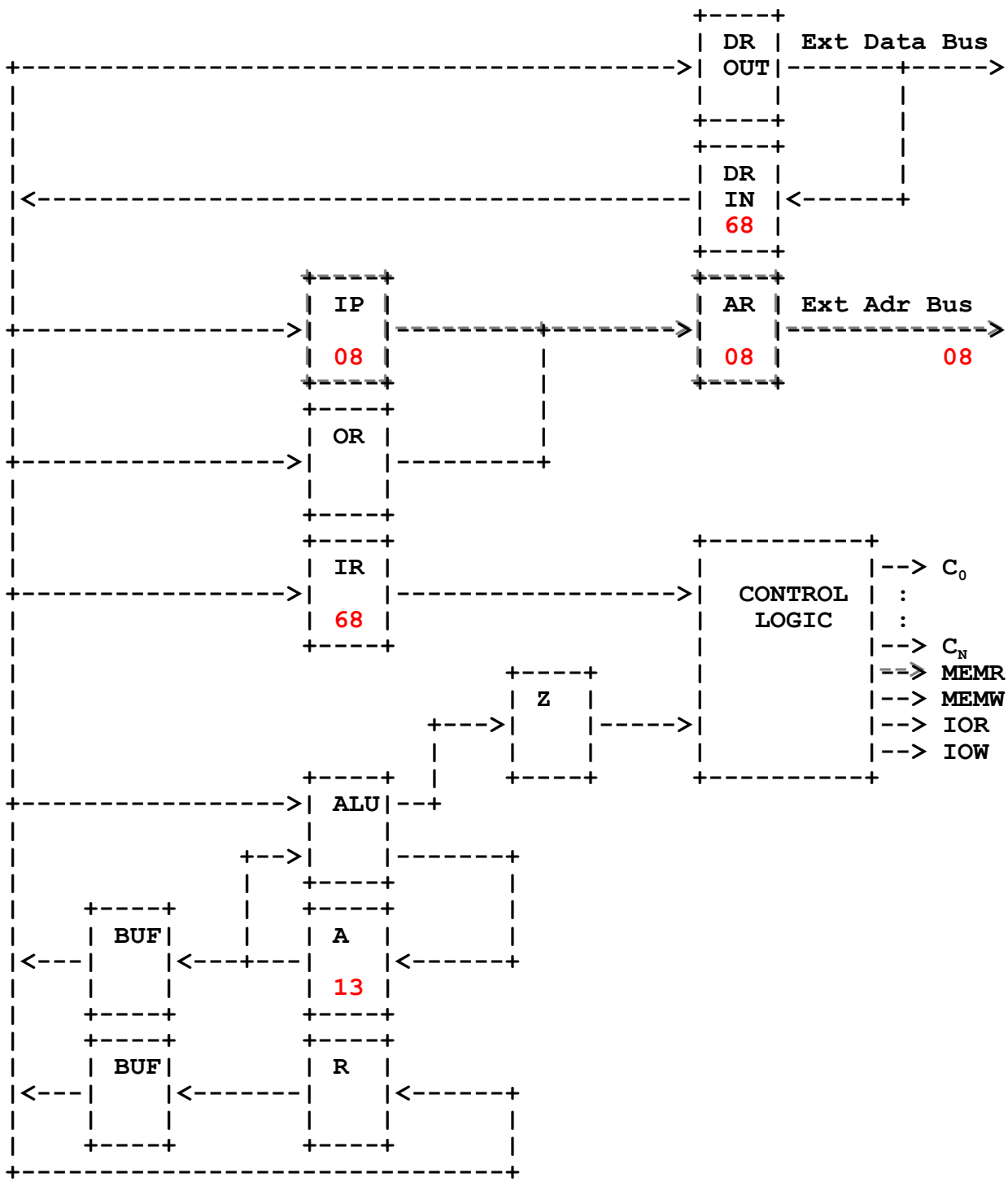


Figure 5.4: Instruction Cycle (Execute 1) for SUB 1Ah

Execute Cycle Step 2

(External Address Bus <-- AR; Assert MEMR; DR <-- Byte # 2;
IP <-- IP + 1)

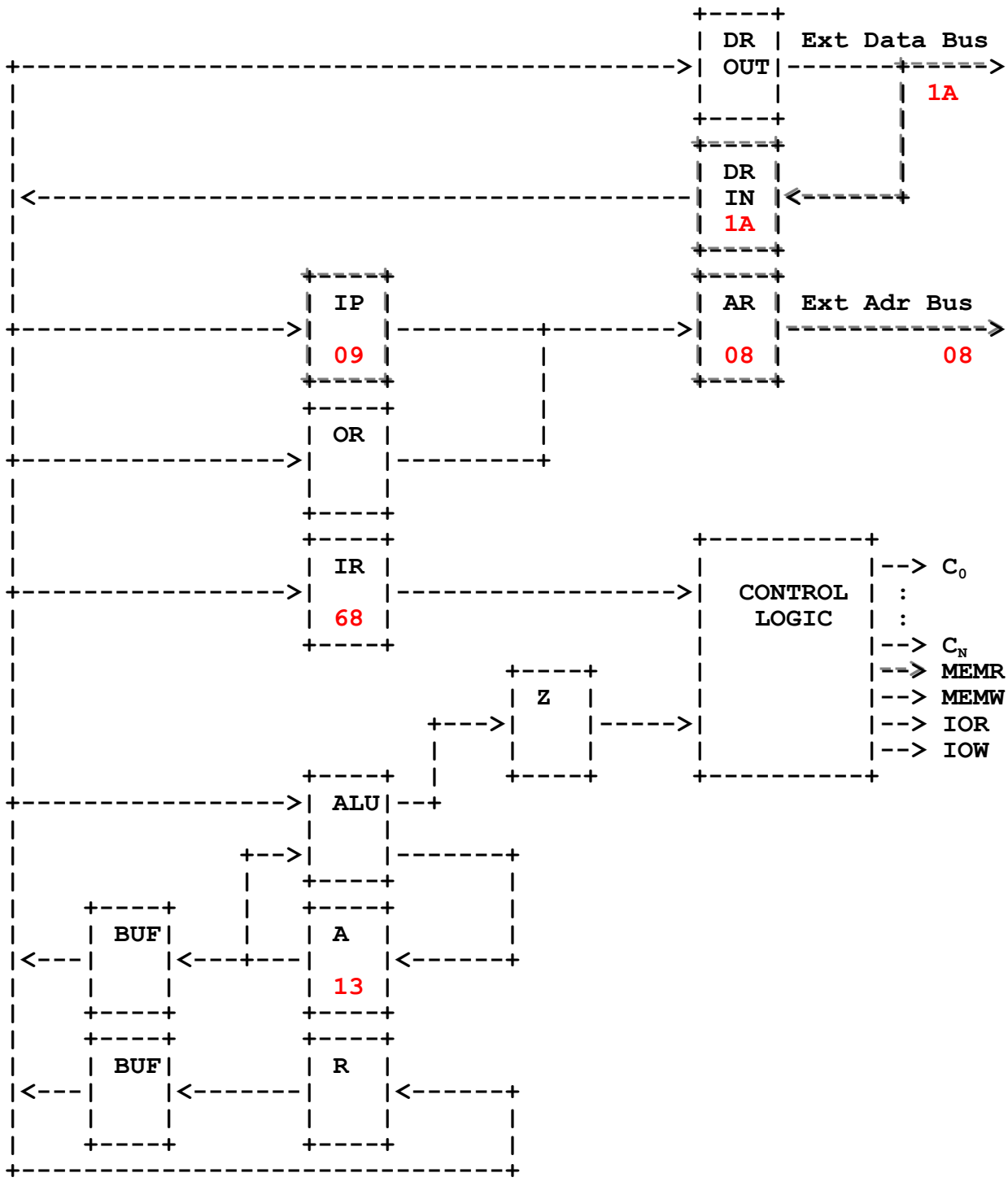


Figure 5.5: Instruction Cycle (Execute 2) for SUB 1Ah

Execute Cycle Step 3

(OR <-- DR)

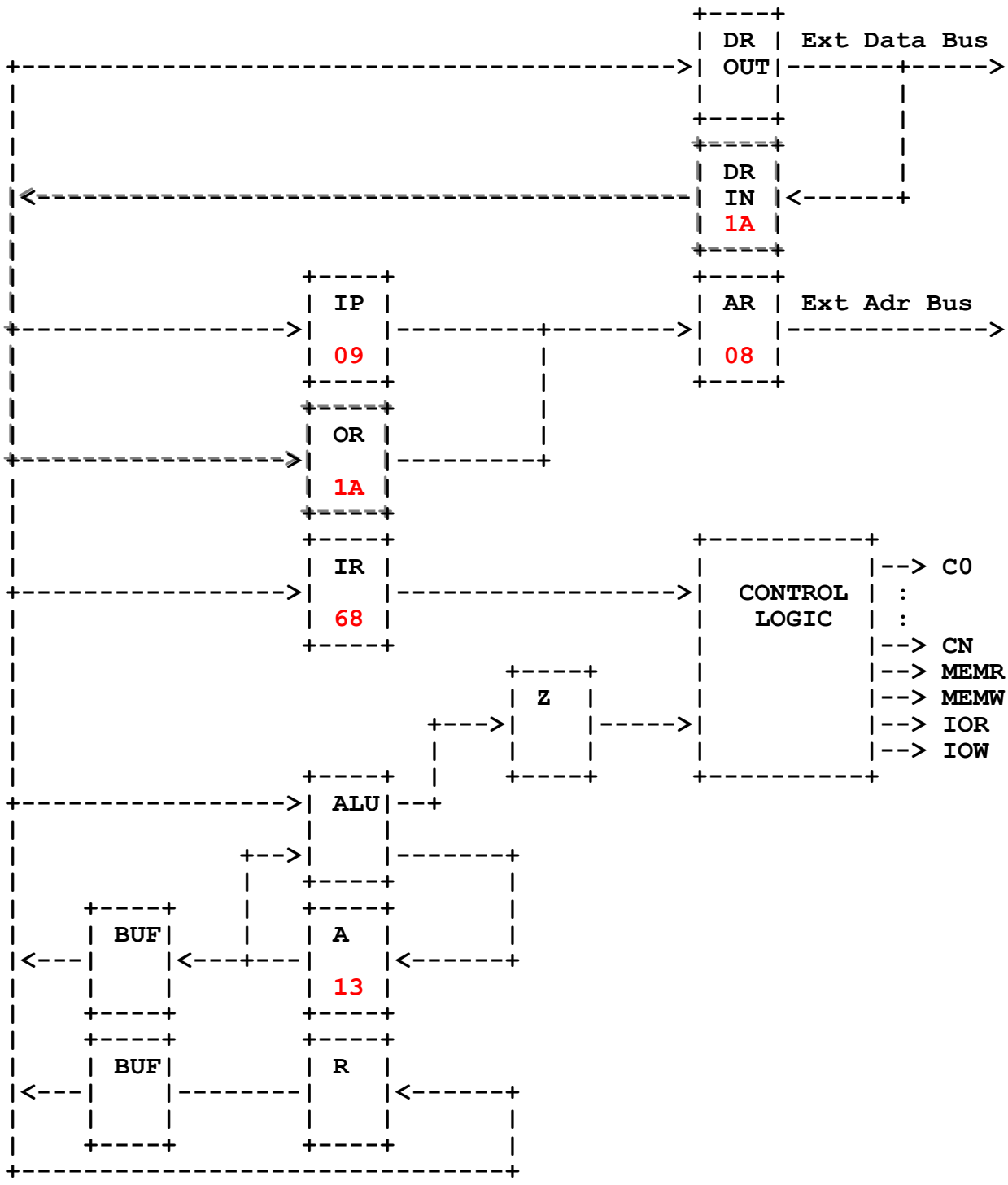


Figure 5.6: Instruction Cycle (Execute 3) for SUB 1Ah

Step 4

The control bits in CS address 683h trigger the following datapath events:

Event	Control Bits
AR <-- OR	!OR_OUT, AR_LOAD
External Address Bus <-- AR	!AR_OUT
Assert MEMR	MEMR

The output of OR is enabled and the 1Ah in OR is placed on the inputs of AR. On CLK2, 1Ah is loaded into AR. The output of AR is enabled and 1Ah is placed on the external address bus. MEMR is asserted. See Figure 5.7. MIP increments the CS address to 684h on CLK1.

Step 5

The control bits in CS address 684h trigger the following datapath events:

Event	Control Bits
External Address Bus <-- AR	!AR_OUT
Assert MEMR	MEMR
DR <-- Operand	DR(IN)_LOAD

The output of AR continues to be enabled and 1Ah remains on the external address bus. MEMR remains asserted. On CLK2, the 11h, which was in memory location 1Ah, is loaded into DR. See Figure 5.8. MIP increments the CS address to 685h on CLK1.

Step 6

The control bits in CS address 685h trigger the following datapath events:

Event	Control Bits
ALU(B) <-- DR	!DR(IN)_OUT
ALU(A) <-- A	Always Present
ALU(F) <-- ALU(A) - ALU(B)	SUB
A <-- ALU(F)	A_LOAD
IR <-- 00h (FETCH)	IR_RESET

The output of DR is enabled and 11h is placed on the internal bus (IB) and the B input of the ALU. The contents of the accumulator are always present on the A input of the ALU. PAL 2 uses control bit 17 (SUB) to encode the four select bits, mode bit, and carry in bit of the ALU to perform a subtract operation. The results are immediately available on the F output of the ALU. On CLK2 the accumulator is loaded with the results of the subtraction. Control bit 0 (IR_RESET), which is "1", is inverted and latched

by PAL 3. The resulting "0" is forwarded, through PAL 2, to the active-low CLR input of the instruction register (IR). This resets IR to 00h (opcode for FETCH). PAL 3 simultaneously latches a "0" on the active-low LOAD of the microinstruction pointer (MIP). See Figure 5.9. Because MIP resets synchronously, nothing happens until CLK1, when MIP is loaded with a 0h. This addresses the first microinstruction of FETCH at CS address 000h and the instruction cycle repeats.

Execute Cycle Step 4

(AR <-- OR; External address bus <-- AR; Assert MEMR)

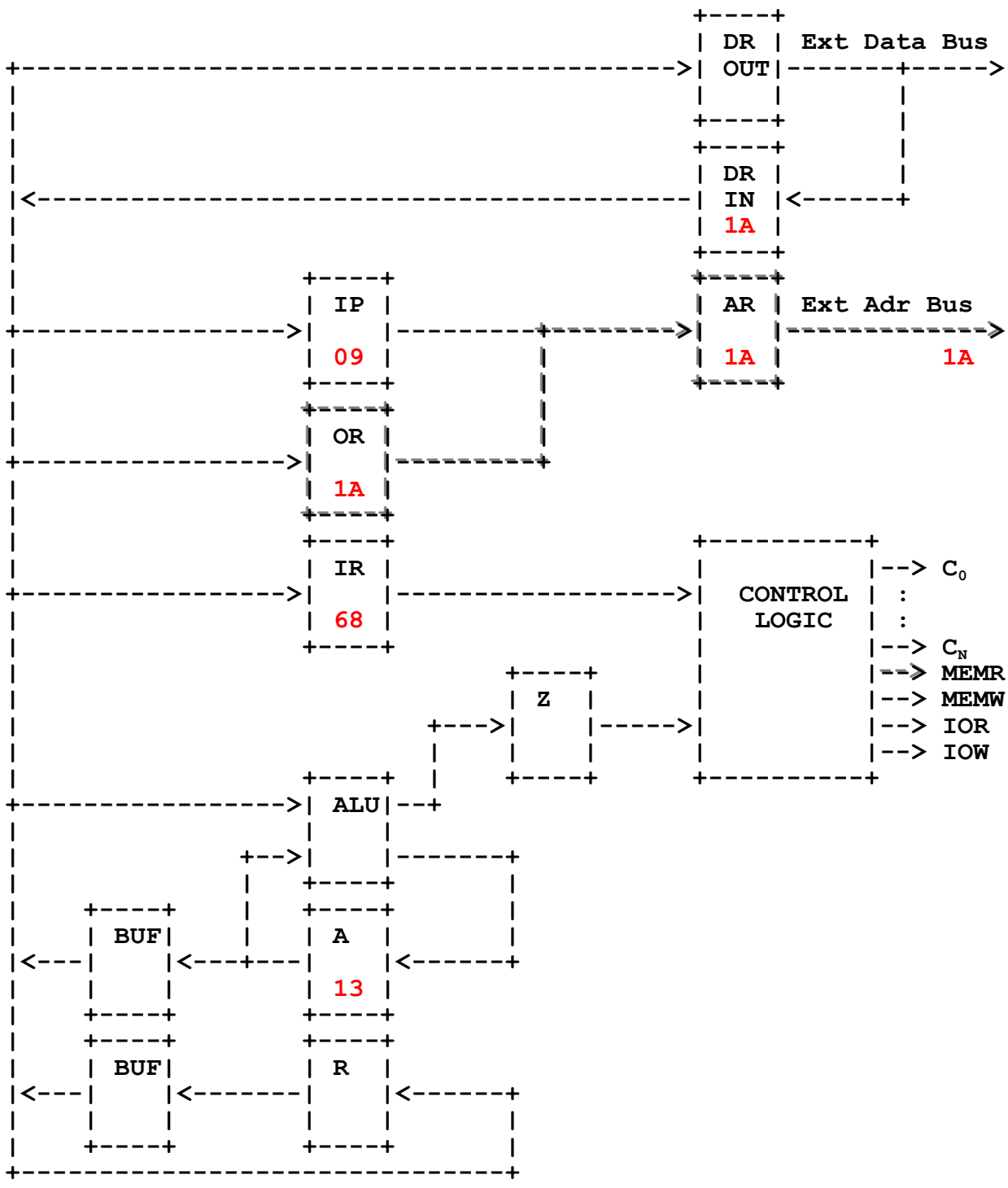


Figure 5.7: Instruction Cycle (Execute 4) for SUB 1Ah

Execute Cycle Step 5

(External Address Bus <-- AR; Assert MEMR; DR <-- Operand)

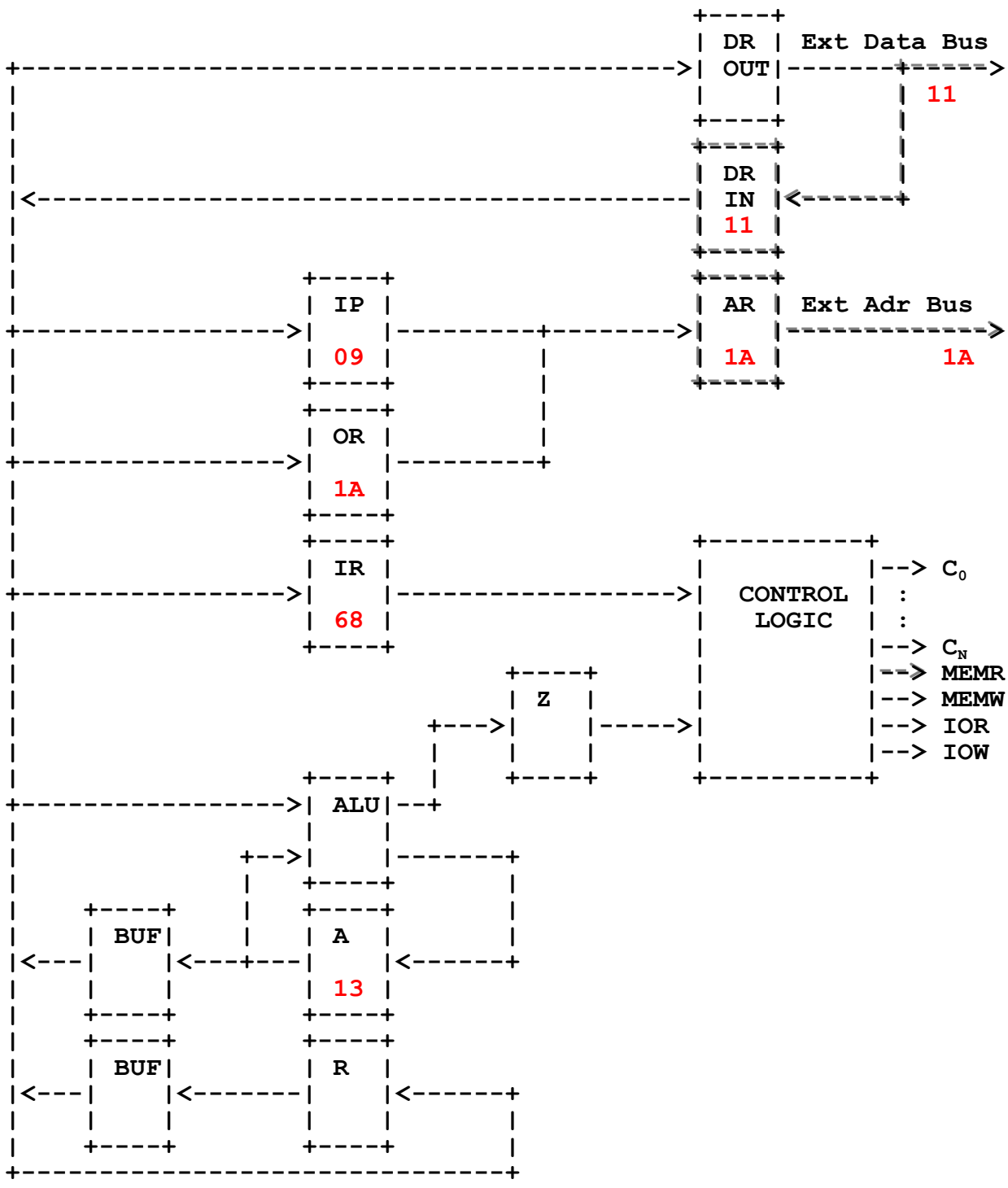


Figure 5.8: Instruction Cycle (Execute 5) for SUB 1Ah

Execute Cycle Step 6

(ALU(B) <-- DR; ALU(A) <-- A; ALU(F) <-- ALU(A) - ALU(B);
A <-- ALU(F))

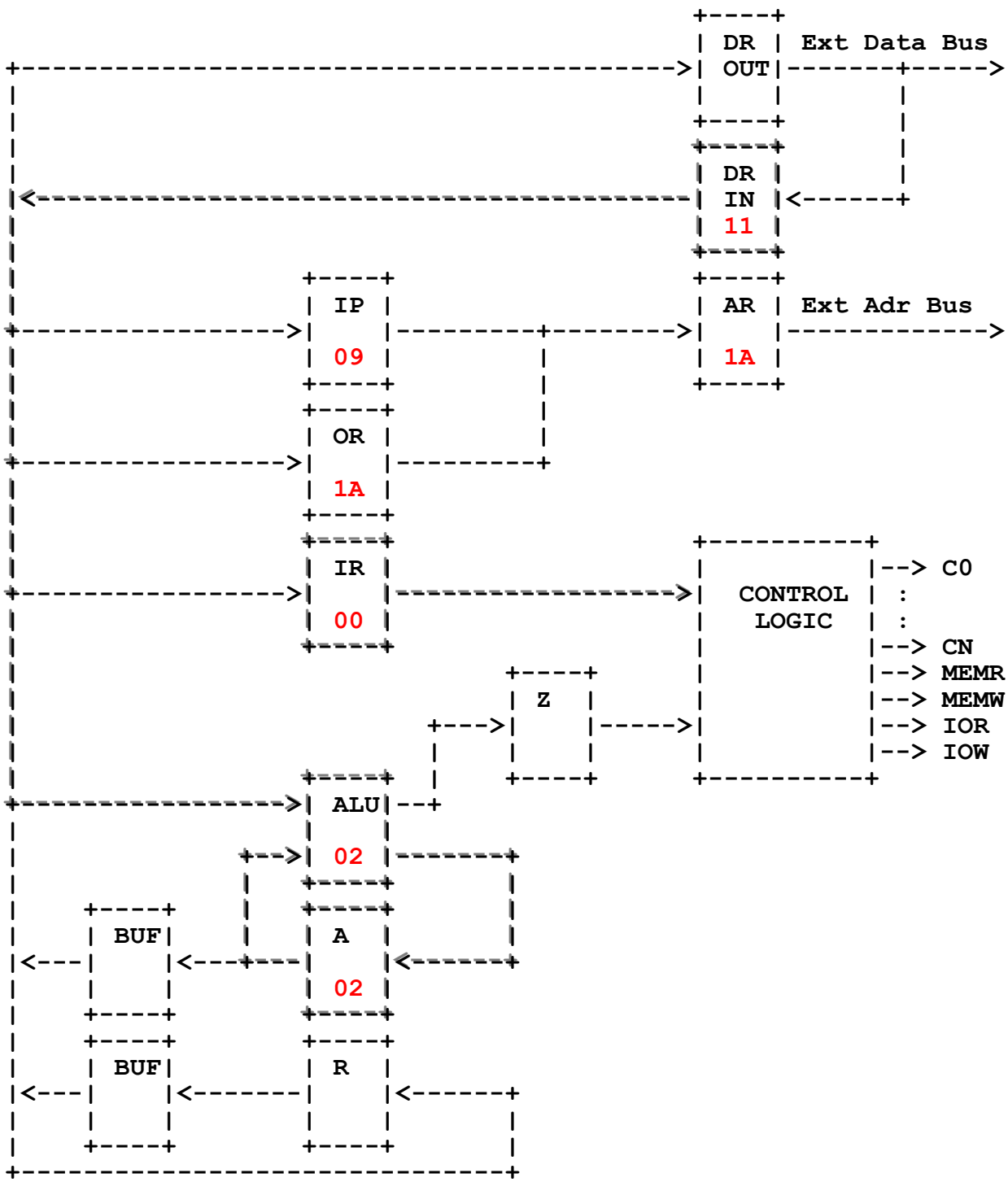


Figure 5.9: Instruction Cycle (Execute 6) for SUB 1Ah

6.0 References

Carter, J. W. (1997). *Digital designing with programmable logic devices.* Upper Saddle River, NJ: Prentice Hall.

Comer, D. J. (1990). *Digital logic and state machine design.* Philadelphia: Saunders.

Haskell, R. E. (1993). *Introduction to computer engineering: logic design and the 8086 microprocessor.* Engle Cliffs, NJ: Prentice Hall.

Hayes, J. P. (1988). *Computer architecture and organization.* New York: McGraw-Hill.

Hennessy, J. L., & Patterson, D. A. (1990). *Computer architecture: a quantitative approach.* San Mateo, CA: Kaufmann.

Heuring, V.P., & Jordan, H. F. (1997). *Computer systems design and architecture.* Menlo Park, CA: Addison-Wesley.

Karalis, E. (1997). *Digital design principles and computer architecture.* Upper Saddle River, NJ: Prentice Hall.

Lynn, D. (1996, Winter Semester). *University of Idaho Engineering Outreach course: EE 441 computer organization.* [Video]. Moscow, ID.

McCalla, T. R. (1992). *Digital logic and computer design.* New York: Merrill.

Miller, M. A. (1997). *Digital devices and systems.* Albany, NY: Delmar.

Pappas, N. L. (1994). *Digital design.* Minneapolis/St. Paul: West.

Streib, W. J. (1997). *Digital circuits.* Tinley Park, IL: Goodheart-Wilcox.

Thompson, A. W. (1995). *Understanding microprocessors: a practical approach.* Albany, NY: Delmar.

(1993). *Programmable logic devices databook and design guide.* Santa Clara, CA: National Semiconductor Corporation.

(1992). *FAST and LS TTL data.* Phoenix, Az: Motorola, Incorporated.

(1990). *Memory*. Mt. Prospect, IL: Intel Literature Sales.

(1983). *PAL handbook.* Santa Clara, CA: Monolithic Memories, Incorporated.

(1983). *CUPL pld/fpga language compiler.* Deerfield Beach, FL: Logical Devices, Incorporated.

(1978). *Logic -- TTL data manual.* Sunnyvale, CA: Signetics Corporation.

7.0 Appendixes

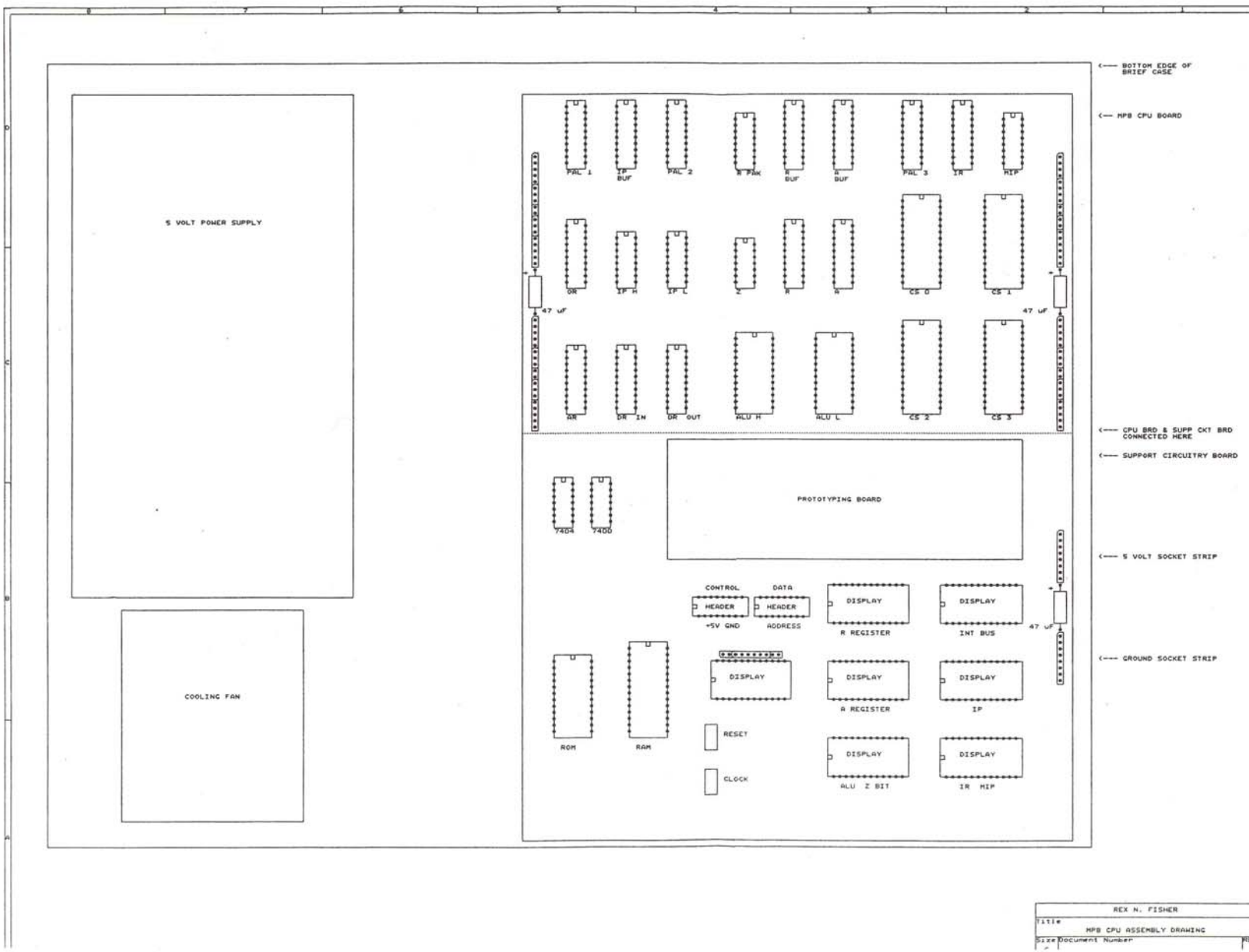
7.1 Appendix: Comprehensive Description of P8 CPU Instruction Set

7.2 Appendix: Microprogram Listing

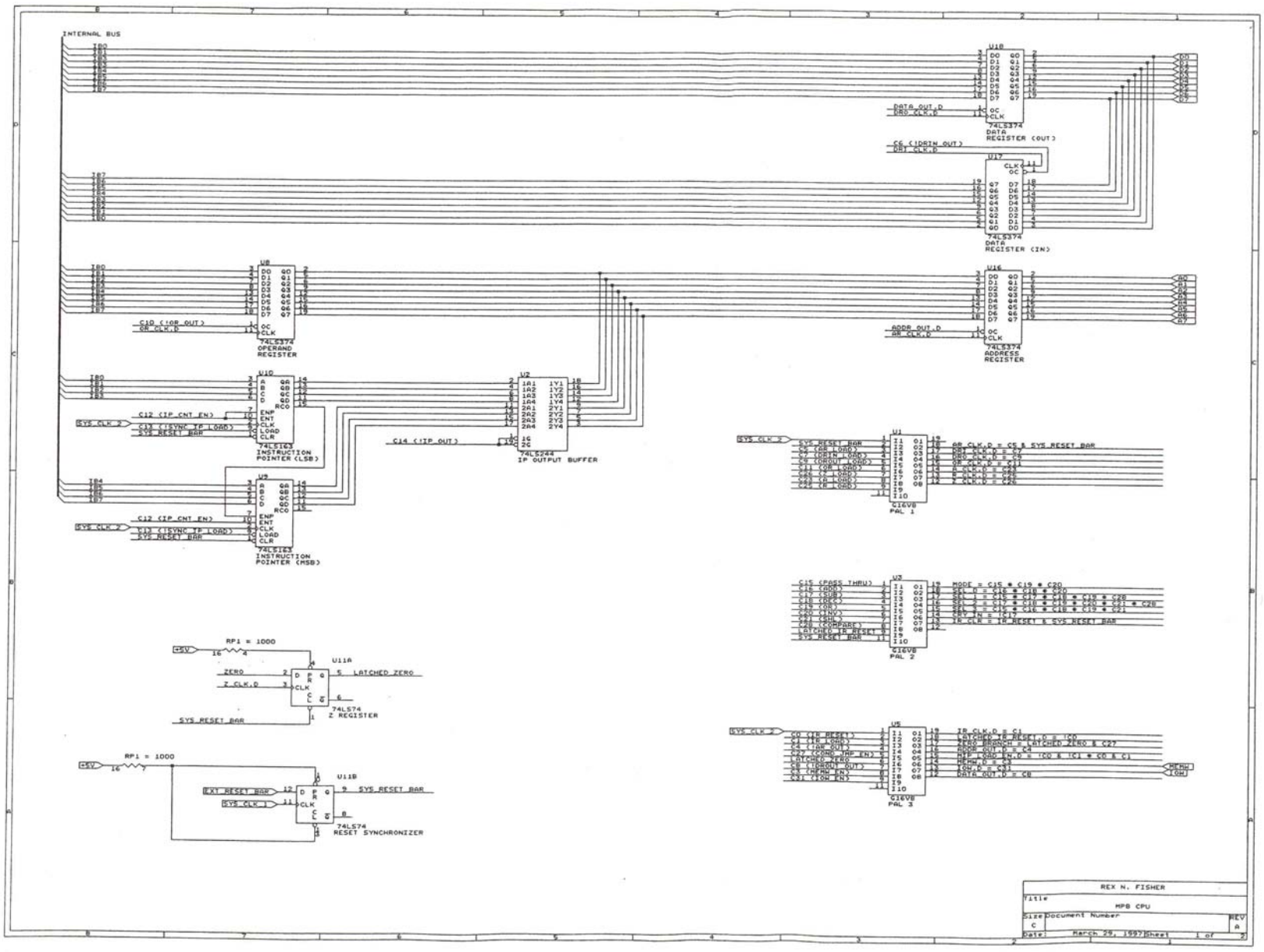
7.3 Appendix: PAL Listings

7.4 Appendix: Schematic Diagrams

7.5 Appendix: Assembly Drawings



REX N. FISHER	
Title	MPB CPU ASSEMBLY DRAWING
Size	Document Number / REV


INTERNAL BUS
U18
74LS374
DATA
REGISTER (OUT)
DATA_OUT_D
DRO_CLK_D
C6 (IDBIN_OUT)
DBI_CLK_D
U17
74LS374
DATA
REGISTER (IN)
U8
74LS374
OPERAND
REGISTER
C10 (IOR_OUT)
OR_CLK_D
U16
74LS374
ADDRESS
REGISTER
ADDR_OUT_D
AR_CLK_D
U10
74LS163
INSTRUCTION
POINTER (LSB)
C12 (IP_CNT_EN)
C13 (ISYNC_IP_LOAD)
SYS_RESET_BAR
SYS_CLK_2
U9
74LS163
INSTRUCTION
POINTER (MSB)
U2
74LS244
IP OUTPUT BUFFER
C14 (IIP_OUT)
U1
C16V8
PAL 1
U7
C16V8
PAL 2
U5
C16V8
PAL 3
RP1 = 1000
+5V
U11A
74LS74
Z REGISTER
ZERO
Z_CLK_D
LATCHED_ZERO
U11B
74LS74
RESET SYNCHRONIZER
EXT_RESET_BAR
REX N. FISHER
Title
MPB CPU
Size
C
Document Number
REV
A
Date: March 29, 1997
Sheet 1 of 2

INTERNAL BUS
ZERO BRANCH
INSTRUCTION REGISTER
A OUTPUT BUFFER
B OUTPUT BUFFER
A REGISTER
B REGISTER
ALU (LSB)
ALU (MSB)
CONTROL STORE 0 (LSB)
CONTROL STORE 1
CONTROL STORE 2
CONTROL STORE 3
ZERO
REX N. FISHER
MPR CPU

```
Name            PAL 1
Designer        Rex N. Fisher;
Assembly        P8 8-Bit CPU;


/* Target Device & Mode */

    /* G16V8 */

        /*
            16V8 Architecture              DIP Pin Count: 20
            Mnemonic: G16V8                 Total Product Terms: 64
        */

    /* Medium Synchronous (Registered) Mode */

        /*
            Input only     Output only    Input/Output
            ----------     -----------    ------------
            2, 3, 4,                      12, 13, 14,
            5, 6, 7,                      15, 16, 17,
            8, 9                          18, 19

            Pin 1 = Common Clock
            Pin 11 = Common Output Enable
        */

    Device  G16V8MS;  /* Designates G16V8 in Registered Mode */


/* Define Logic Operators */

    /* AND = &  */
    /* OR = #   */
    /* NOT = !  */


/* Define Output Pins */

    pin 18 = ar_clk;   /* address register clock input */
    pin 17 = dri_clk;  /* data register (in) clock input */
    pin 16 = dro_clk;  /* data register (out) clock input */
    pin 15 = or_clk;   /* operand register clock input */
    pin 14 = a_clk;    /* a register clock */
    pin 13 = r_clk;    /* r register clock */
    pin 12 = z_clk;    /* z (zero status) register clock */


/* Define Input Pins */

    pin 1 = clk2;         /* phase 2 of clock */
    pin 2 = reset;        /* external reset */
    pin 3 = c5;           /* control store bit 5 */
    pin 4 = c7;           /* control store bit 7 */
    pin 5 = c9;           /* control store bit 9 */
    pin 6 = c11;          /* control store bit 11 */
    pin 7 = c26;          /* control store bit 26 */
    pin 8 = c23;          /* control store bit 23 */
    pin 9 = c25;          /* control store bit 25 */
    pin 11 = !oe;         /* output enable -- ground this pin */
```

```
/* Boolean Equations */

     ar_clk.d = c5 & reset;
     dri_clk.d = c7;
     dro_clk.d = c9;
     or_clk.d = c11;
     a_clk.d = c23;
     r_clk.d = c25;
     z_clk.d = c26;
```

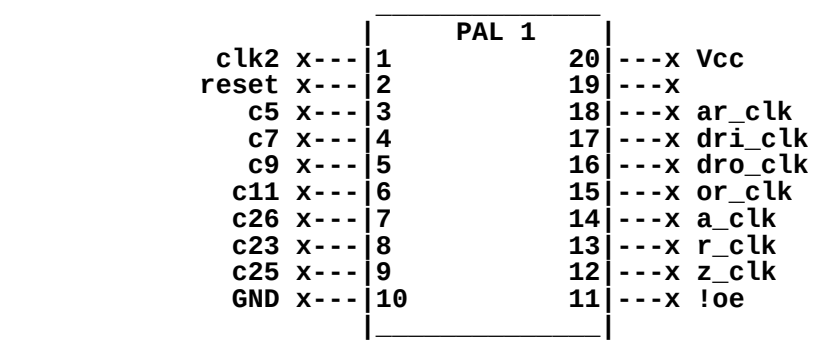
```
********************************************************************************
                                    PAL 1
********************************************************************************

CUPLPLD         4.2a Serial# MD-22410301
Device          g16v8ms  Library DLIB-h-82-11
Created         Fri Mar 28 13:00:50 1997
Name            PAL 1
Designer        Rex N. Fisher

================================================================================
                          Expanded Product Terms
================================================================================

a_clk.d  =>
    c23

ar_clk.d  =>
    c5 & reset

dri_clk.d  =>
    c7

dro_clk.d  =>
    c9

or_clk.d  =>
    c11

r_clk.d  =>
    c25

z_clk.d  =>
    c26
```

```
================================================================================
                                  Symbol Table
================================================================================

Pin Variable                                          Pterms   Max     Min
Pol   Name                 Ext     Pin     Type       Used   Pterms  Level
--- --------               ---     ---     ----      ------  ------  -----

    a_clk                          14       V          -       -       -
    a_clk                  d       14       X          1       8       4
    ar_clk                         18       V          -       -       -
    ar_clk                 d       18       X          1       8       4
    c5                             3        V          -       -       -
    c7                             4        V          -       -       -
    c9                             5        V          -       -       -
    c11                            6        V          -       -       -
    c23                            8        V          -       -       -
    c25                            9        V          -       -       -
    c26                            7        V          -       -       -
    clk2                           1        V          -       -       -
    dri_clk                        17       V          -       -       -
    dri_clk                d       17       X          1       8       4
    dro_clk                        16       V          -       -       -
    dro_clk                d       16       X          1       8       4
 !  oe                             11       V          -       -       -
    or_clk                         15       V          -       -       -
    or_clk                 d       15       X          1       8       4
    r_clk                          13       V          -       -       -
    r_clk                  d       13       X          1       8       4
    reset                          2        V          -       -       -
    z_clk                          12       V          -       -       -
    z_clk                  d       12       X          1       8       4


LEGEND    D : default variable         F : field      G : group
          I : intermediate variable    N : node       M : extended node
          U : undefined                V : variable   X : extended variable
          T : function
```

```
================================================================================
                                    Fuse Plot
================================================================================

Syn   02192 x Ac0   02193 -

Pin #19  02048  Pol x  02120  Ac1 -
 00000 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00032 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00064 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00096 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00128 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00160 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00192 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00224 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Pin #18  02049  Pol -  02121  Ac1 x
 00256 x---x--------------------------
 00288 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00320 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00352 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00384 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00416 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00448 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00480 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Pin #17  02050  Pol -  02122  Ac1 x
 00512 --------x-------------------------
 00544 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00576 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00608 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00640 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00672 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00704 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00736 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Pin #16  02051  Pol -  02123  Ac1 x
 00768 ------------x-----------------------
 00800 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00832 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00864 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00896 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00928 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00960 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00992 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Pin #15  02052  Pol -  02124  Ac1 x
 01024 ----------------x---------------
 01056 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01088 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01120 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01152 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01184 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01216 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01248 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Pin #14  02053  Pol -  02125  Ac1 x
 01280 ------------------------x-------
 01312 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01344 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01376 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01408 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01440 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01472 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01504 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Pin #13  02054  Pol -  02126  Ac1 x
```

```
 01536 ----------------------------x---
 01568 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01600 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01632 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01664 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01696 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01728 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01760 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Pin #12  02055  Pol -  02127  Ac1 x
 01792 --------------------x-----------
 01824 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01856 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01888 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01920 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01952 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01984 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 02016 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx


LEGEND    X : fuse not blown
          - : fuse blown
```

Chip Diagram



```
Name            PAL 2
Designer        Rex N. Fisher;
Assembly        P8 8-Bit CPU;


/* Target Device & Mode */

    /* G16V8 */

         /*
             16V8 Architecture              DIP Pin Count: 20
             Mnemonic: G16V8                 Total Product Terms: 64
         */

    /* Simple (Small) Mode */

         /*
             Input only     Output only    Input/Output
             ----------     -----------    ------------
             1, 2, 3,         15, 16       12, 13, 14,
             4, 5, 6,                      17, 18, 19
             7, 8, 9,
             11
         */

    Device  G16V8S;  /* Designates G16V8 in Simple Mode */


/* Define Logic Operators */

    /* AND = &  */
    /* OR = #   */
    /* NOT = !  */


/* Define Output Pins */

    pin 19 = mode;       /* alu "mode" input */
    pin 18 = sel_0;      /* alu "s0" input */
    pin 17 = sel_1;      /* alu "s1" input */
    pin 16 = sel_2;      /* alu "s2" input */
    pin 15 = sel_3;      /* alu "s3" input */
    pin 14 = cry_in;     /* alu "cn" input */
    pin 13 = ir_clr;     /* clear input for instruction register */


/* Define Input Pins */

    pin 1 = c15;         /* control store bit 15 (pass_thru) */
    pin 2 = c16;         /* control store bit 16 (add) */
    pin 3 = c17;         /* control store bit 17 (sub) */
    pin 4 = c18;         /* control store bit 18 (dec) */
    pin 5 = c19;         /* control store bit 19 (or) */
    pin 6 = c20;         /* control store bit 20 (inv) */
    pin 7 = c21;         /* control store bit 21 (shl) */
    pin 8 = c28;         /* control store bit 28 (compare) */
    pin 9 = ir_reset;    /* latched inv cs bit 0, input from PAL 3 */
    pin 11 = sys_reset_bar;   /* external system reset */
```

```
/* Boolean Equations */

     mode = c15 # c19 # c20;
     sel_0 = c16 # c18 # c20;
     sel_1 = c15 # c17 # c18 # c19 # c28;
     sel_2 = c17 # c18 # c19 # c20 # c21 # c28;
     sel_3 = c15 # c16 # c18 # c19 # c21;
     cry_in = !c17;
     ir_clr = ir_reset & sys_reset_bar;
```

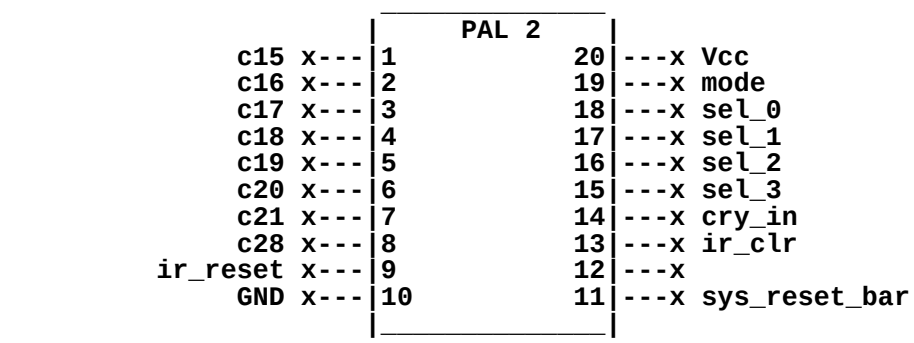
```
********************************************************************************
                                      PAL 2
********************************************************************************

CUPLPLD         4.2a Serial# MD-22410301
Device          g16v8s  Library DLIB-h-82-9
Created         Thu Mar 27 17:27:27 1997
Name            PAL 2
Designer        Rex N. Fisher

================================================================================
                            Expanded Product Terms
================================================================================

cry_in =>
    !c17

ir_clr =>
    ir_reset & sys_reset_bar

mode =>
    c20
  # c19
  # c15

sel_0 =>
    c20
  # c18
  # c16

sel_1 =>
    c28
  # c19
  # c18
  # c17
  # c15

sel_2 =>
    c28
  # c21
  # c20
  # c19
  # c18
  # c17

sel_3 =>
    c21
  # c19
  # c18
  # c16
  # c15
```

```
================================================================================
                                Symbol Table
================================================================================

Pin Variable                                          Pterms   Max     Min
Pol   Name              Ext     Pin     Type         Used   Pterms  Level
--- --------            ---     ---     ----        ------  ------  -----

    c15                         1        V           -       -       -
    c16                         2        V           -       -       -
    c17                         3        V           -       -       -
    c18                         4        V           -       -       -
    c19                         5        V           -       -       -
    c20                         6        V           -       -       -
    c21                         7        V           -       -       -
    c28                         8        V           -       -       -
    cry_in                      14       V           1       8       4
    ir_clr                      13       V           1       8       4
    ir_reset                    9        V           -       -       -
    mode                        19       V           3       8       4
    sel_0                       18       V           3       8       4
    sel_1                       17       V           5       8       4
    sel_2                       16       V           6       8       4
    sel_3                       15       V           5       8       4
    sys_reset_bar               11       V           -       -       -

LEGEND    D : default variable          F : field      G : group
          I : intermediate variable     N : node       M : extended node
          U : undefined                 V : variable   X : extended variable
          T : function
```

```
================================================================================
                                    Fuse Plot
================================================================================

Syn   02192 - Ac0   02193 x

Pin #19  02048  Pol -  02120  Ac1 x
 00000 ----------------x--------------
 00032 ------------x------------------
 00064 --x----------------------------
 00096 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00128 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00160 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00192 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00224 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Pin #18  02049  Pol -  02121  Ac1 x
 00256 ----------------x--------------
 00288 --------x----------------------
 00320 x----------------------------
 00352 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00384 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00416 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00448 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00480 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Pin #17  02050  Pol -  02122  Ac1 x
 00512 ------------------------x-------
 00544 ------------x------------------
 00576 --------x----------------------
 00608 ----x--------------------------
 00640 --x----------------------------
 00672 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00704 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00736 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Pin #16  02051  Pol -  02123  Ac1 x
 00768 ------------------------x-------
 00800 --------------------x-----------
 00832 ----------------x--------------
 00864 ------------x------------------
 00896 --------x----------------------
 00928 ----x--------------------------
 00960 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00992 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Pin #15  02052  Pol -  02124  Ac1 x
 01024 --------------------x-----------
 01056 ------------x------------------
 01088 --------x----------------------
 01120 x----------------------------
 01152 --x----------------------------
 01184 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01216 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01248 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Pin #14  02053  Pol -  02125  Ac1 x
 01280 -----x-------------------------
 01312 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01344 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01376 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01408 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01440 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01472 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01504 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Pin #13  02054  Pol -  02126  Ac1 x
```

```
 01536 ----------------------------x-x-
 01568 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01600 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01632 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01664 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01696 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01728 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01760 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Pin #12  02055  Pol x  02127  Ac1 -
 01792 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01824 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01856 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01888 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01920 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01952 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01984 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 02016 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx


LEGEND    X : fuse not blown
          - : fuse blown
```

Chip Diagram



```
Name            PAL 3
Designer        Rex N. Fisher;
Assembly        P8 8-Bit CPU;


/* Target Device & Mode */

    /* G16V8 */

         /*
             16V8 Architecture              DIP Pin Count: 20
             Mnemonic: G16V8                 Total Product Terms: 64
         */

    /* Medium Synchronous (Registered) Mode */

         /*
             Input only     Output only    Input/Output
             ----------     -----------    ------------
             2, 3, 4,                      12, 13, 14,
             5, 6, 7,                      15, 16, 17,
             8, 9                          18, 19

             Pin 1 = Common Clock
             Pin 11 = Common Output Enable
         */

    Device  G16V8MS;  /* Designates G16V8 in Registered Mode */


/* Define Logic Operators */

    /* AND = &  */
    /* OR = #   */
    /* NOT = !  */


/* Define Output Pins */

    pin 19 = ir_clk;         /* load instruction register (ir) */
    pin 18 = ir_reset;       /* latched cs bit 0, input for PAL 2 */
    pin 17 = zero_branch;    /* branch address bit (a4) for jnz & jz */
    pin 16 = addr_out;       /* ar register output enable */
    pin 15 = mip_load_en;    /* enable (sync) parallel load of mip */
    pin 14 = memw;           /* memory write bit */
    pin 13 = iow;            /* i/o write bit */
    pin 12 = data_out;       /* dr (out) register output enable */


/* Define Input Pins */

    pin 1 = clk2;            /* phase 2 of clock */
    pin 2 = c0;              /* control store bit 0 (ir_reset) */
    pin 3 = c1;              /* control store bit 1 (ir_load) */
    pin 4 = c4;              /* control store bit 4 (!ar_out) */
    pin 5 = c27;             /* control store bit 27 (cond_jmp_en) */
    pin 6 = latched_zero;    /* output from z register */
    pin 7 = c8;              /* control store bit 8 (!drout_out) */
    pin 8 = c3;              /* control store bit 3 (memw_en) */
    pin 9 = c31;             /* control store bit 31 (iow_en) */
    pin 11 = !oe;            /* output enable -- ground this pin */
```

```
/* Boolean Equations */

     ir_clk.d = c1;
     ir_reset.d = !c0;
     zero_branch = latched_zero & c27;
     addr_out.d = c4;
     mip_load_en.d = !c0 & !c1 # c0 & c1;
     memw.d = c3;
     iow.d = c31;
     data_out.d = c8;
```

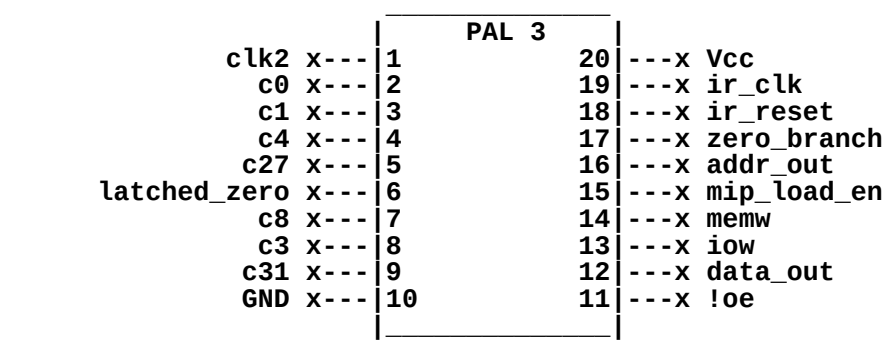
```
********************************************************************************
                                      PAL 3
********************************************************************************

CUPLPLD         4.2a Serial# MD-22410301
Device          g16v8ms  Library DLIB-h-82-11
Created         Thu Mar 27 17:28:41 1997
Name            PAL 3
Designer        Rex N. Fisher

================================================================================
                            Expanded Product Terms
================================================================================

addr_out.d  =>
    c4

data_out.d  =>
    c8

iow.d  =>
    c31

ir_clk.d  =>
    c1

ir_reset.d  =>
    !c0

memw.d  =>
    c3

mip_load_en.d  =>
    !c0 & !c1
  # c0 & c1

zero_branch =>
    c27 & latched_zero

zero_branch.oe  =>
    1
```

```
================================================================================
                                Symbol Table
================================================================================

Pin Variable                                          Pterms   Max     Min
Pol   Name              Ext     Pin     Type          Used   Pterms  Level
--- --------            ---     ---     ----          ------ ------  -----

    addr_out                    16       V             -      -       -
    addr_out            d       16       X             1      8       4
    c0                          2        V             -      -       -
    c1                          3        V             -      -       -
    c3                          8        V             -      -       -
    c4                          4        V             -      -       -
    c8                          7        V             -      -       -
    c27                         5        V             -      -       -
    c31                         9        V             -      -       -
    clk2                        1        V             -      -       -
    data_out                    12       V             -      -       -
    data_out            d       12       X             1      8       4
    iow                         13       V             -      -       -
    iow                 d       13       X             1      8       4
    ir_clk                      19       V             -      -       -
    ir_clk              d       19       X             1      8       4
    ir_reset                    18       V             -      -       -
    ir_reset            d       18       X             1      8       4
    latched_zero                6        V             -      -       -
    memw                        14       V             -      -       -
    memw                d       14       X             1      8       4
    mip_load_en                 15       V             -      -       -
    mip_load_en         d       15       X             2      8       4
 !  oe                          11       V             -      -       -
    zero_branch                 17       V             1      7       4
    zero_branch         oe      17       D             1      1       0


LEGEND    D : default variable         F : field      G : group
          I : intermediate variable    N : node       M : extended node
          U : undefined                V : variable   X : extended variable
          T : function
```

```
================================================================================
                                    Fuse Plot
================================================================================

Syn   02192 x Ac0   02193 -

Pin #19  02048  Pol -  02120  Ac1 x
 00000 ----x--------------------------
 00032 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00064 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00096 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00128 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00160 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00192 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00224 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Pin #18  02049  Pol -  02121  Ac1 x
 00256 -x-----------------------------
 00288 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00320 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00352 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00384 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00416 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00448 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00480 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Pin #17  02050  Pol -  02122  Ac1 -
 00512 -------------------------------
 00544 ------------x---x--------------
 00576 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00608 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00640 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00672 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00704 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00736 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Pin #16  02051  Pol -  02123  Ac1 x
 00768 --------x-----------------------
 00800 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00832 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00864 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00896 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00928 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00960 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00992 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Pin #15  02052  Pol -  02124  Ac1 x
 01024 -x---x--------------------------
 01056 x---x--------------------------
 01088 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01120 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01152 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01184 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01216 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01248 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Pin #14  02053  Pol -  02125  Ac1 x
 01280 ----------------------x-------
 01312 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01344 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01376 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01408 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01440 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01472 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01504 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Pin #13  02054  Pol -  02126  Ac1 x
```

```
 01536 ----------------------------x---
 01568 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01600 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01632 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01664 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01696 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01728 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01760 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Pin #12  02055  Pol -  02127  Ac1 x
 01792 --------------------x-----------
 01824 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01856 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01888 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01920 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01952 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01984 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 02016 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx


LEGEND    X : fuse not blown
          - : fuse blown
```

Chip Diagram



The control store is composed of four 2764 PROMs. Only 12 of the 13 address bits are used. PROMs with only 12 address bits would have worked, but none were available for this project. Four PROMs provide a total of 32 data bits that can be used to control the CPU. Only 31 control bits are required by the P8.

Each control store address is 12 bits long. The eight most significant bits (A_{11} - A_4) are set by the instruction register IR), which contains the 8-bit opcode of the instruction being executed. The opcode of each instruction specifies a block of 16 addresses within the control store where the microwords for its execution are located.

The four least significant bits (A_3 - A_0) select the individual microwords required by the opcode in the instruction register. The microinstruction pointer (MIP) is a binary counter that drives address bits A_2, A_1, and A_0. This allows each instruction to have as many as eight microwords. Address bit A_3 is used only by the conditional jumps. The decision on whether to execute the conditional jump depends on the state of A_3.

See Figure 7.2.1 for a block diagram that shows how the control bits are addressed.

The microprogram contained in the control store is listed in this appendix. Each page shows an MP8 instruction, the addresses of its microwords, and the control bits that comprise the microinstructions. The addresses are divided into four fields:

OP: This is the 5-bit operation code that identifies each of the MP8 instruction types.

ADR: This is the 3-bit code that identifies the addressing mode and register used by each instruction. There are as many as five different codes for each instruction type identified by OP.

The OP and ADR fields are combined to form the 8-bit opcode unique to each instruction. See Appendix 7.1.

Z: This is A_3, which is used by conditional jump instructions to determine whether the jump should be executed. It is high during a conditional jump when the condition bit (Zero) is set. It is always low otherwise.

MIP: This is the 3-bit output of the MIP. It sequences through each micro-instruction until it is reset by the last one.

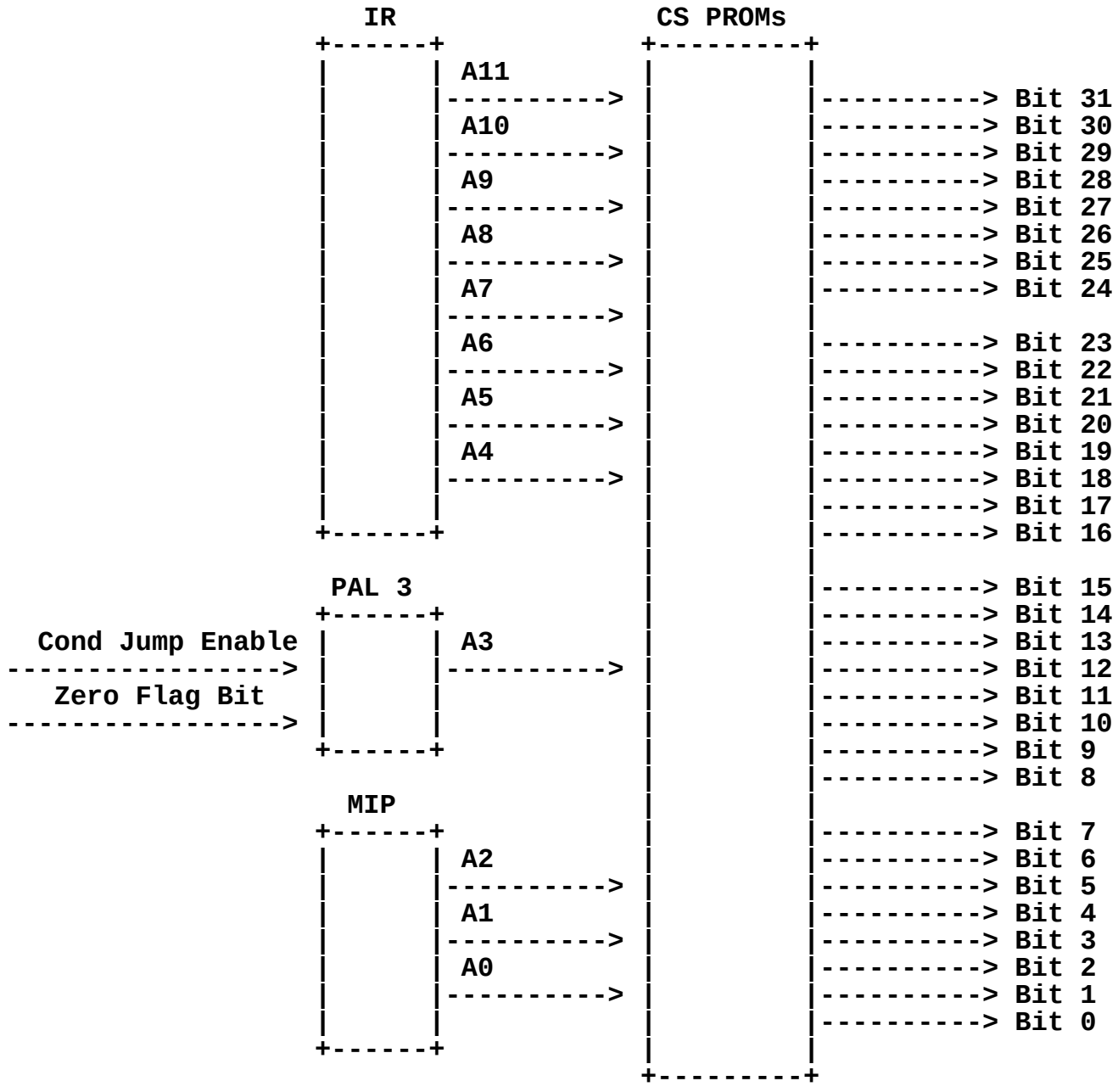


Figure 7.2.1: Block Diagram of Control Store Addressing

FETCH

Address				Data																															
				IOW	IOR	SPARE	COMPARE	COND JMP	ZLOAD	RLOAD	!ROUT	ALOAD	!AOUT	SHL	INV	OR	DEC	SUB	ADD	PASSTHRU	!IPOUT	!SYNC IPLOAD	IPCLKEN	ORLOAD	!OROUT	DROUTLOAD	!DROUT	DRINLOAD	!DRINOUT	ARLOAD	!AROUT	MEMW	MEMR	IRLOAD	IRRESET
OP	ADR	Z	MIP	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
00000	000	0	000	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	1	0	0	1	0	1	0	1	1	0	0	1	0	0
		0	001	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	1	0	1	0	1	1	1	0	0	0	1	0	0
		0	010	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	0	0	1	0	1	0	0	0	1	0	0	1	0
		0	011																																
		0	100																																
		0	101																																
		0	110																																
		0	111																																
		1	000	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	1	0	0	1	0	1	0	1	1	0	0	1	0	0
		1	001																																
		1	010																																
		1	011																																
		1	100																																
		1	101																																
		1	110																																
		1	111																																

IN Address

Address **Data**

OP	ADR	Z	MIP	IOW 31	IOR 30	SPARE 29	COMPARE 28	COND JMP EN 27	ZLOAD 26	RLOAD 25	!ROUT 24	ALOAD 23	!AOUT 22	SHL 21	INV 20	OR 19	DEC 18	SUB 17	ADD 16	PASTHRU 15	!IPOUT 14	!SYNC IPLOAD 13	IPCLKEN 12	ORLOAD 11	!OROUT 10	DROUTLOAD 09	!DROUT 08	DRINLOAD 07	!DRINOUT 06	ARLOAD 05	!AROUT 04	MEMW 03	MEMR 02	IRLOAD 01	IRRESET 00
00001	000	0	000	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	1	0	0	1	0	1	0	1	1	0	0	1	0	0
		0	001	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	1	0	1	0	1	1	1	0	0	0	1	0	0
		0	010	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	0	1	1	0	1	0	0	0	1	0	0	0	0
		0	011	0	1	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	0	0	0	0	1	0	1	1	0	0	0	0	0
		0	100	0	1	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	0	0	1	0	1	1	1	0	0	0	0	0	0
		0	101	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	1	1	1	0	0	1	0	1	0	0	0	1	0	0	0	1
		0	110																																
		0	111																																
		1	000																																
		1	001																																
		1	010																																
		1	011																																
		1	100																																
		1	101																																
		1	110																																
		1	111																																

IN P

Address **Data**

OP	ADR	Z	MIP	IOW 31	IOR 30	SPARE 29	COMPARE 28	COND JMP EN 27	ZLOAD 26	RLOAD 25	!ROUT 24	ALOAD 23	!AOUT 22	SHL 21	INV 20	OR 19	DEC 18	SUB 17	ADD 16	PASSTHRU 15	!IPOUT 14	!SYNC IPLOAD 13	IPCLKEN 12	ORLOAD 11	!OROUT 10	DROUTLOAD 09	!DROUT 08	DRINLOAD 07	!DRINOUT 06	ARLOAD 05	!AROUT 04	MEMW 03	MEMR 02	IRLOAD 01	IRRESET 00
00001	100	0	000	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	1	0	1	1	0	1	0	1	0	1	0	0	0	0
		0	001	0	1	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	0	0	0	0	1	0	1	1	0	0	0	0	0
		0	010	0	1	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	0	0	1	0	1	1	1	0	0	0	0	0	0
		0	011	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	1	1	1	0	0	1	0	1	0	0	0	1	0	0	0	1
		0	100																																
		0	101																																
		0	110																																
		0	111																																
		1	000																																
		1	001																																
		1	010																																
		1	011																																
		1	100																																
		1	101																																
		1	110																																
		1	111																																

OUT Address

Address **Data**

				IOW	IOR	SPARE	COMPARE	COND JMP	ZLOAD	RLOAD	!ROUT	ALOAD	!AOUT	SHL	INV	OR	DEC	SUB	ADD	PASSTHRU	!IPOUT	!SYNC IPLOAD	IPCLKEN	ORLOAD	!OROUT	DROUTLOAD	!DROUT	DRINLOAD	!DRINOUT	ARLOAD	!AROUT	MEMW	MEMR	IRLOAD	IRRESET
OP	ADR	Z	MIP	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
00010	000	0	000	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	1	0	0	1	0	1	0	1	1	0	0	1	0	0
		0	001	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	1	0	1	0	1	1	1	0	0	0	1	0	0
		0	010	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	0	1	1	0	1	0	0	0	1	0	0	0	0
		0	011	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	1	1	0	0	0	0	0
		0	100	1	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	0	0	1	0	0	0	1	0	0	0	0	0	0
		0	101	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	0	0	1	0	0	0	1	0	0	0	0	0	1
		0	110																																
		0	111																																
		1	000																																
		1	001																																
		1	010																																
		1	011																																
		1	100																																
		1	101																																
		1	110																																
		1	111																																

OUT P

Address **Data**

OP	ADR	Z	MIP	31 IOW	30 IOR	29 SPARE	28 COMPARE	27 COND JMP EN	26 ZLOAD	25 RLOAD	24 !ROUT	23 ALOAD	22 !AOUT	21 SHL	20 INV	19 OR	18 DEC	17 SUB	16 ADD	15 PASS THRU	14 !IPOUT	13 !SYNC IPLOAD	12 IPCLK EN	11 ORLOAD	10 !OROUT	09 DROUT LOAD	08 !DROUT OUT	07 DRIN LOAD	06 !DRIN OUT	05 ARLOAD	04 !AROUT	03 MEMW	02 MEMR	01 IRLOAD	00 IR RESET
00010	100	0	000	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	1	0	1	1	0	1	0	1	0	1	0	0	0	0
		0	001	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	1	1	0	0	0	0	0
		0	010	1	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	0	0	1	0	0	0	1	0	0	0	0	0	0
		0	011	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	0	0	1	0	0	0	1	0	0	0	0	0	1
		0	100																																
		0	101																																
		0	110																																
		0	111																																
		1	000																																
		1	001																																
		1	010																																
		1	011																																
		1	100																																
		1	101																																
		1	110																																
		1	111																																

JMP Address

Address **Data**

OP	ADR	Z	MIP	31 IOW	30 IOR	29 SPARE	28 COMPARE	27 COND JMP EN	26 ZLOAD	25 RLOAD	24 !ROUT	23 ALOAD	22 !AOUT	21 SHL	20 INV	19 OR	18 DEC	17 SUB	16 ADD	15 PASSTHRU	14 !IPOUT	13 !SYNC IPLOAD	12 IPCLKEN	11 ORLOAD	10 !OROUT	09 DROUTLOAD	08 !DROUTOUT	07 DRINLOAD	06 !DRINOUT	05 ARLOAD	04 !AROUT	03 MEMW	02 MEMR	01 IRLOAD	00 IRRESET
00100	000	0	000	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	1	0	0	1	0	1	0	1	1	0	0	1	0	0
		0	001	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	0	0	1	0	1	1	1	0	0	0	1	0	0
		0	010	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	0	1	0	1	0	1	0	0	0	1	0	0	0	1
		0	011																																
		0	100																																
		0	101																																
		0	110																																
		0	111																																
		1	000																																
		1	001																																
		1	010																																
		1	011																																
		1	100																																
		1	101																																
		1	110																																
		1	111																																

JMP R

Address **Data**

OP	ADR	Z	MIP	31 IOW	30 IOR	29 SPARE	28 COMPARE	27 COND JMP EN	26 ZLOAD	25 RLOAD	24 !ROUT	23 ALOAD	22 !AOUT	21 SHL	20 INV	19 OR	18 DEC	17 SUB	16 ADD	15 PASSTHRU	14 !IPOUT	13 !SYNC IPLOAD	12 IPCLKEN	11 ORLOAD	10 !OROUT	09 DROUTLOAD	08 !DROUTOUT	07 DRINLOAD	06 !DRINOUT	05 ARLOAD	04 !AROUT	03 MEMW	02 MEMR	01 IRLOAD	00 IRRESET
00100	011	0	000	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	0	0	1
		0	001																																
		0	010																																
		0	011																																
		0	100																																
		0	101																																
		0	110																																
		0	111																																
		1	000																																
		1	001																																
		1	010																																
		1	011																																
		1	100																																
		1	101																																
		1	110																																
		1	111																																

JNZ Address

Address **Data**

OP	ADR	Z	MIP	31 IOW	30 IOR	29 SPARE	28 COMPARE	27 COND JMP EN	26 ZLOAD	25 RLOAD	24 !ROUT	23 ALOAD	22 !AOUT	21 SHL	20 INV	19 OR	18 DEC	17 SUB	16 ADD	15 PASSTHRU	14 !IPOUT	13 !SYNC IPLOAD	12 IPCLKEN	11 ORLOAD	10 !OROUT	09 DROUTLOAD	08 !DROUTOUT	07 DRINLOAD	06 !DRINOUT	05 ARLOAD	04 !AROUT	03 MEMW	02 MEMR	01 IRLOAD	00 IRRESET
00101	000	0	000	0	0	0	0	1	0	0	1	0	1	0	0	0	0	0	0	0	0	1	0	0	1	0	1	0	1	1	0	0	1	0	0
		0	001	0	0	0	0	1	0	0	1	0	1	0	0	0	0	0	0	0	1	1	0	0	1	0	1	1	1	0	0	0	1	0	0
		0	010	0	0	0	0	1	0	0	1	0	1	0	0	0	0	0	0	0	1	0	1	0	1	0	1	0	0	0	1	0	0	0	1
		0	011																																
		0	100																																
		0	101																																
		0	110																																
		0	111																																
		1	000	0	0	0	0	1	0	0	1	0	1	0	0	0	0	0	0	0	1	1	1	0	1	0	1	0	1	0	1	0	0	0	1
		1	001																																
		1	010																																
		1	011																																
		1	100																																
		1	101																																
		1	110																																
		1	111																																

JNZ R

Address **Data**

OP	ADR	Z	MIP	31 IOW	30 IOR	29 SPARE	28 COMPARE	27 COND JMP EN	26 ZLOAD	25 RLOAD	24 !ROUT	23 ALOAD	22 !AOUT	21 SHL	20 INV	19 OR	18 DEC	17 SUB	16 ADD	15 PASSTHRU	14 !IPOUT	13 !SYNC IPLOAD	12 IPCLKEN	11 ORLOAD	10 !OROUT	09 DROUTLOAD	08 !DROUTOUT	07 DRINLOAD	06 !DRINOUT	05 ARLOAD	04 !AROUT	03 MEMW	02 MEMR	01 IRLOAD	00 IRRESET
00101	011	0	000	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	0	0	1
		0	001																																
		0	010																																
		0	011																																
		0	100																																
		0	101																																
		0	110																																
		0	111																																
		1	000	0	0	0	0	1	0	0	1	0	1	0	0	0	0	0	0	0	1	1	0	0	1	0	1	0	1	0	1	0	0	0	1
		1	001																																
		1	010																																
		1	011																																
		1	100																																
		1	101																																
		1	110																																
		1	111																																

JZ Address

Address				Data																																			
				IOW	IOR	SPARE	COMPARE	COND JMP	ZLOAD	RLOAD	!ROUT	ALOAD	!AOUT	SHL	INV	OR	DEC	SUB	ADD	PASSTHRU	!IPOUT	!SYNC IPLOAD	IPCLKEN	ORLOAD	!OROUT	DROUTLOAD	!DROUTOUT	DRINLOAD	!DRINOUT	ARLOAD	!AROUT	MEMW	MEMR	IRLOAD	IRRESET				
OP	ADR	Z	MIP	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00				
00110	000	0	000	0	0	0	0	1	0	0	1	0	1	0	0	0	0	0	0	0	1	1	1	0	1	0	1	0	1	0	1	0	0	0	1				
		0	001																																				
		0	010																																				
		0	011																																				
		0	100																																				
		0	101																																				
		0	110																																				
		0	111																																				
		1	000	0	0	0	0	1	0	0	1	0	1	0	0	0	0	0	0	0	0	1	0	0	1	0	1	0	1	1	0	0	1	0	0				
		1	001	0	0	0	0	1	0	0	1	0	1	0	0	0	0	0	0	0	1	1	0	0	1	0	1	1	1	0	0	0	1	0	0				
		1	010	0	0	0	0	1	0	0	1	0	1	0	0	0	0	0	0	0	1	0	1	0	1	0	1	0	0	0	1	0	0	0	1				
		1	011																																				
		1	100																																				
		1	101																																				
		1	110																																				
		1	111																																				

JZ R

Address **Data**

OP	ADR	Z	MIP	IOW 31	IOR 30	SPARE 29	COMPARE 28	COND JMP EN 27	ZLOAD 26	RLOAD 25	!ROUT 24	ALOAD 23	!AOUT 22	SHL 21	INV 20	OR 19	DEC 18	SUB 17	ADD 16	PASSTHRU 15	!IPOUT 14	!SYNC IPLOAD 13	IPCLKEN 12	ORLOAD 11	!OROUT 10	DROUTLOAD 09	!DROUT 08	DRINLOAD 07	!DRINOUT 06	ARLOAD 05	!AROUT 04	MEMW 03	MEMR 02	IRLOAD 01	IRRESET 00
00000	000	0	000	0	0	0	0	1	0	0	1	0	1	0	0	0	0	0	0	0	1	1	0	0	1	0	1	0	1	0	1	0	0	0	1
		0	001																																
		0	010																																
		0	011																																
		0	100																																
		0	101																																
		0	110																																
		0	111																																
		1	000	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	0	0	1
		1	001																																
		1	010																																
		1	011																																
		1	100																																
		1	101																																
		1	110																																
		1	111																																

CMP Address

Address **Data**

OP	ADR	Z	MIP	IOW 31	IOR 30	SPARE 29	COMPARE 28	CND JMP 27	ZLOAD 26	RLOAD 25	!ROUT 24	ALOAD 23	!AOUT 22	SHL 21	INV 20	OR 19	DEC 18	SUB 17	ADD 16	PASTHRU 15	!IPOUT 14	!SYNC IPLOAD 13	IPCLKEN 12	ORLOAD 11	!OROUT 10	DROUTLOAD 09	!DROUT 08	DRINLOAD 07	!DRINOUT 06	ARLOAD 05	!AROUT 04	MEMW 03	MEMR 02	IRLOAD 01	IRRESET 00
00111	000	0	000	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	1	0	0	1	0	1	0	1	1	0	0	1	0	0
		0	001	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	1	0	1	0	1	1	1	0	0	0	1	0	0
		0	010	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	0	1	1	0	1	0	0	0	1	0	0	0	0
		0	011	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	0	0	0	0	1	0	1	1	0	0	1	0	0
		0	100	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	0	0	1	0	1	1	1	0	0	0	1	0	0
		0	101	0	0	0	1	0	1	0	1	0	1	0	0	0	0	0	0	0	1	1	0	0	1	0	1	0	0	0	1	0	0	0	1
		0	110																																
		0	111																																
		1	000																																
		1	001																																
		1	010																																
		1	011																																
		1	100																																
		1	101																																
		1	110																																
		1	111																																

CMP A

Address **Data**

OP	ADR	Z	MIP	31 IOW	30 IOR	29 SPARE	28 COMPARE	27 COND JMP EN	26 ZLOAD	25 RLOAD	24 !ROUT	23 ALOAD	22 !AOUT	21 SHL	20 INV	19 OR	18 DEC	17 SUB	16 ADD	15 PASSTHRU	14 !IPOUT	13 !SYNC IPLOAD	12 IPCLKEN	11 ORLOAD	10 !OROUT	09 DROUTLOAD	08 !DROUTOUT	07 DRINLOAD	06 !DRINOUT	05 ARLOAD	04 !AROUT	03 MEMW	02 MEMR	01 IRLOAD	00 IRRESET
00111	010	0	000	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0	1	0	1	0	1	0	0	0	1
		0	001																																
		0	010																																
		0	011																																
		0	100																																
		0	101																																
		0	110																																
		0	111																																
		1	000																																
		1	001																																
		1	010																																
		1	011																																
		1	100																																
		1	101																																
		1	110																																
		1	111																																

CMP R

Address **Data**

OP	ADR	Z	MIP	31 IOW	30 IOR	29 SPARE	28 COMPARE	27 COND JMP EN	26 ZLOAD	25 RLOAD	24 !ROUT	23 ALOAD	22 !AOUT	21 SHL	20 INV	19 OR	18 DEC	17 SUB	16 ADD	15 PASSTHRU	14 !IPOUT	13 !SYNC IPLOAD	12 IPCLKEN	11 ORLOAD	10 !OROUT	09 DROUTLOAD	08 !DROUTOUT	07 DRINLOAD	06 !DRINOUT	05 ARLOAD	04 !AROUT	03 MEMW	02 MEMR	01 IRLOAD	00 IRRESET
00111	011	0	000	0	0	0	1	0	1	0	0	0	1	0	0	0	0	0	0	0	1	1	0	0	1	0	1	0	1	0	1	0	0	0	1
		0	001																																
		0	010																																
		0	011																																
		0	100																																
		0	101																																
		0	110																																
		0	111																																
		1	000																																
		1	001																																
		1	010																																
		1	011																																
		1	100																																
		1	101																																
		1	110																																
		1	111																																

CMP M

Address **Data**

OP	ADR	Z	MIP	IOW	IOR	SPARE	COMPARE	COND JMP	ZLOAD	RLOAD	!ROUT	ALOAD	!AOUT	SHL	INV	OR	DEC	SUB	ADD	PASSTHRU	!IPOUT	!SYNC IPLOAD	IPCLKEN	ORLOAD	!OROUT	DROUTLOAD	!DROUTOUT	DRINLOAD	!DRINOUT	ARLOAD	!AROUT	MEMW	MEMR	IRLOAD	IRRESET
				31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
00111	100	0	000	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	1	0	1	1	0	1	0	1	0	1	0	0	0	0
		0	001	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	0	0	0	0	1	0	1	1	0	0	1	0	0
		0	010	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	0	0	1	0	1	1	1	0	0	0	1	0	0
		0	011	0	0	0	1	0	1	0	1	0	1	0	0	0	0	0	0	0	1	1	0	0	1	0	1	0	0	0	1	0	0	0	1
		0	100																																
		0	101																																
		0	110																																
		0	111																																
		1	000																																
		1	001																																
		1	010																																
		1	011																																
		1	100																																
		1	101																																
		1	110																																
		1	111																																

CMP I Data

Address				Data																															
				IOW	IOR	SPARE	COMPARE	CONDJMP	ZLOAD	RLOAD	!ROUT	ALOAD	!AOUT	SHL	INV	OR	DEC	SUB	ADD	PASSTHRU	!IPOUT	!SYNCIPLOAD	IPCLKEN	ORLOAD	!OROUT	DROUTLOAD	!DROUTOUT	DRINLOAD	!DRINOUT	ARLOAD	!AROUT	MEMW	MEMR	IRLOAD	IRRESET
OP	ADR	Z	MIP	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
00111	110	0	000	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	1	0	0	1	0	1	0	1	1	0	0	1	0	0
		0	001	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	1	0	1	0	1	1	1	0	0	0	1	0	0
		0	010	0	0	0	1	0	1	0	1	0	1	0	0	0	0	0	0	0	1	1	0	0	1	0	1	0	0	0	1	0	0	0	1
		0	011																																
		0	100																																
		0	101																																
		0	110																																
		0	111																																
		1	000																																
		1	001																																
		1	010																																
		1	011																																
		1	100																																
		1	101																																
		1	110																																
		1	111																																

LDA Address

Address **Data**

OP	ADR	Z	MIP	IOW 31	IOR 30	SPARE 29	COMPARE 28	COND JMP EN 27	ZLOAD 26	RLOAD 25	!ROUT 24	ALOAD 23	!AOUT 22	SHL 21	INV 20	OR 19	DEC 18	SUB 17	ADD 16	PASSTHRU 15	!IPOUT 14	!SYNC IPLOAD 13	IPCLKEN 12	ORLOAD 11	!OROUT 10	DROUTLOAD 09	!DROUT 08	DRINLOAD 07	!DRINOUT 06	ARLOAD 05	!AROUT 04	MEMW 03	MEMR 02	IRLOAD 01	IRRESET 00
01000	000	0	000	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	1	0	0	1	0	1	0	1	1	0	0	1	0	0
		0	001	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	1	0	1	0	1	1	1	0	0	0	1	0	0
		0	010	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	0	1	1	0	1	0	0	0	1	0	0	0	0
		0	011	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	0	0	0	0	1	0	1	1	0	0	1	0	0
		0	100	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	0	0	1	0	1	1	1	0	0	0	1	0	0
		0	101	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	1	1	1	0	0	1	0	1	0	0	0	1	0	0	0	1
		0	110																																
		0	111																																
		1	000																																
		1	001																																
		1	010																																
		1	011																																
		1	100																																
		1	101																																
		1	110																																
		1	111																																

LDA A

Address **Data**

OP	ADR	Z	MIP	IOW 31	IOR 30	SPARE 29	COMPARE 28	COND JMP EN 27	ZLOAD 26	RLOAD 25	!ROUT 24	ALOAD 23	!AOUT 22	SHL 21	INV 20	OR 19	DEC 18	SUB 17	ADD 16	PASSTHRU 15	!IPOUT 14	!SYNC IPLOAD 13	IPCLKEN 12	OR LOAD 11	!OROUT 10	DROUT LOAD 09	!DROUT 08	DRIN LOAD 07	!DRINOUT 06	ARLOAD 05	!AROUT 04	MEMW 03	MEMR 02	IRLOAD 01	IRRESET 00
01000	010	0	000	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	0	0	1	0	1	0	1	0	1	0	0	0	1
		0	001																																
		0	010																																
		0	011																																
		0	100																																
		0	101																																
		0	110																																
		0	111																																
		1	000																																
		1	001																																
		1	010																																
		1	011																																
		1	100																																
		1	101																																
		1	110																																
		1	111																																

LDA R

Address **Data**

OP	ADR	Z	MIP	IOW 31	IOR 30	SPARE 29	COMPARE 28	COND JMP EN 27	ZLOAD 26	RLOAD 25	!ROUT 24	ALOAD 23	!AOUT 22	SHL 21	INV 20	OR 19	DEC 18	SUB 17	ADD 16	PASSTHRU 15	!IPOUT 14	!SYNC IPLOAD 13	IPCLKEN 12	ORLOAD 11	!OROUT 10	DROUTLOAD 09	!DROUT 08	DRINLOAD 07	!DRINOUT 06	ARLOAD 05	!AROUT 04	MEMW 03	MEMR 02	IRLOAD 01	IRRESET 00
01000	011	0	000	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	1	1	0	0	1	0	1	0	1	0	1	0	0	0	1
		0	001																																
		0	010																																
		0	011																																
		0	100																																
		0	101																																
		0	110																																
		0	111																																
		1	000																																
		1	001																																
		1	010																																
		1	011																																
		1	100																																
		1	101																																
		1	110																																
		1	111																																

LDA M

Address **Data**

OP	ADR	Z	MIP	31 IOW	30 IOR	29 SPARE	28 COMPARE	27 COND JMP EN	26 ZLOAD	25 RLOAD	24 !ROUT	23 ALOAD	22 !AOUT	21 SHL	20 INV	19 OR	18 DEC	17 SUB	16 ADD	15 PASSTHRU	14 !IPOUT	13 !SYNC IPLOAD	12 IPCLKEN	11 ORLOAD	10 !OROUT	09 DROUTLOAD	08 !DROUTOUT	07 DRINLOAD	06 !DRINOUT	05 ARLOAD	04 !AROUT	03 MEMW	02 MEMR	01 IRLOAD	00 IRRESET
01000	100	0	000	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	1	0	1	1	0	1	0	1	0	1	0	0	0	0
		0	001	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	0	0	0	0	1	0	1	1	0	0	1	0	0
		0	010	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	0	0	1	0	1	1	1	0	0	0	1	0	0
		0	011	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	1	1	1	0	0	1	0	1	0	0	0	1	0	0	0	1
		0	100																																
		0	101																																
		0	110																																
		0	111																																
		1	000																																
		1	001																																
		1	010																																
		1	011																																
		1	100																																
		1	101																																
		1	110																																
		1	111																																

LDA I Data

OP	ADR	Z	MIP	31 IOW	30 IOR	29 SPARE	28 COMPARE	27 COND JMP EN	26 ZLOAD	25 RLOAD	24 !ROUT	23 ALOAD	22 !AOUT	21 SHL	20 INV	19 OR	18 DEC	17 SUB	16 ADD	15 PASS THRU	14 !IP OUT	13 !SYNC IP LOAD	12 IP CLK EN	11 OR LOAD	10 !OR OUT	09 DROUT LOAD	08 !DROUT OUT	07 DRIN LOAD	06 !DRIN OUT	05 AR LOAD	04 !AR OUT	03 MEMW	02 MEMR	01 IR LOAD	00 IR RESET
01000	110	0	000	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	1	0	0	1	0	1	0	1	1	0	0	1	0	0
		0	001	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	1	0	1	0	1	1	1	0	0	0	1	0	0
		0	010	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	1	1	1	0	0	1	0	1	0	0	0	1	0	0	0	1
		0	011																																
		0	100																																
		0	101																																
		0	110																																
		0	111																																
		1	000																																
		1	001																																
		1	010																																
		1	011																																
		1	100																																
		1	101																																
		1	110																																
		1	111																																

LDR Address

Address **Data**

				IOW	IOR	SPARE	COMPARE	COND JMP EN	ZLOAD	RLOAD	!ROUT	ALOAD	!AOUT	SHL	INV	OR	DEC	SUB	ADD	PASSTHRU	!IPOUT	!SYNC IPLOAD	IPCLKEN	ORLOAD	!OROUT	DROUTLOAD	!DROUT	DRINLOAD	!DRINOUT	ARLOAD	!AROUT	MEMW	MEMR	IRLOAD	IRRESET
OP	ADR	Z	MIP	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
01001	000	0	000	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	1	0	0	1	0	1	0	1	1	0	0	1	0	0
		0	001	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	1	0	1	0	1	1	1	0	0	0	1	0	0
		0	010	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	0	1	1	0	1	0	0	0	1	0	0	0	0
		0	011	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	0	0	0	0	1	0	1	1	0	0	1	0	0
		0	100	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	0	0	1	0	1	1	1	0	0	0	1	0	0
		0	101	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0	0	1	1	1	0	0	1	0	1	0	0	0	1	0	0	0	1
		0	110																																
		0	111																																
		1	000																																
		1	001																																
		1	010																																
		1	011																																
		1	100																																
		1	101																																
		1	110																																
		1	111																																

LDR A

OP	ADR	Z	MIP	31 IOW	30 IOR	29 SPARE	28 COMPARE	27 COND JMP EN	26 ZLOAD	25 RLOAD	24 !ROUT	23 ALOAD	22 !AOUT	21 SHL	20 INV	19 OR	18 DEC	17 SUB	16 ADD	15 PASS THRU	14 !IP OUT	13 !SYNC IP LOAD	12 IP CLK EN	11 OR LOAD	10 !OR OUT	09 DROUT LOAD	08 !DROUT OUT	07 DRIN LOAD	06 !DRIN OUT	05 AR LOAD	04 !AR OUT	03 MEMW	02 MEMR	01 IR LOAD	00 IR RESET
01001	010	0	000	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0	1	0	1	0	1	0	0	0	1
		0	001																																
		0	010																																
		0	011																																
		0	100																																
		0	101																																
		0	110																																
		0	111																																
		1	000																																
		1	001																																
		1	010																																
		1	011																																
		1	100																																
		1	101																																
		1	110																																
		1	111																																

LDR R

Address **Data**

OP	ADR	Z	MIP	31 IOW	30 IOR	29 SPARE	28 COMPARE	27 COND JMP EN	26 ZLOAD	25 RLOAD	24 !ROUT	23 ALOAD	22 !AOUT	21 SHL	20 INV	19 OR	18 DEC	17 SUB	16 ADD	15 PASSTHRU	14 !IPOUT	13 !SYNC IPLOAD	12 IPCLKEN	11 ORLOAD	10 !OROUT	09 DROUTLOAD	08 !DROUTOUT	07 DRINLOAD	06 !DRINOUT	05 ARLOAD	04 !AROUT	03 MEMW	02 MEMR	01 IRLOAD	00 IR RESET
01001	011	0	000	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	0	0	1	0	1	0	1	0	1	0	0	0	1
		0	001																																
		0	010																																
		0	011																																
		0	100																																
		0	101																																
		0	110																																
		0	111																																
		1	000																																
		1	001																																
		1	010																																
		1	011																																
		1	100																																
		1	101																																
		1	110																																
		1	111																																

LDR M

Address **Data**

OP	ADR	Z	MIP	31 IOW	30 IOR	29 SPARE	28 COMPARE	27 COND JMP EN	26 ZLOAD	25 RLOAD	24 !ROUT	23 ALOAD	22 !AOUT	21 SHL	20 INV	19 OR	18 DEC	17 SUB	16 ADD	15 PASSTHRU	14 !IPOUT	13 !SYNC IPLOAD	12 IPCLKEN	11 ORLOAD	10 !OROUT	09 DROUTLOAD	08 !DROUTOUT	07 DRINLOAD	06 !DRINOUT	05 ARLOAD	04 !AROUT	03 MEMW	02 MEMR	01 IRLOAD	00 IR RESET
01001	100	0	000	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	1	0	1	1	0	1	0	1	0	1	0	0	0	0
		0	001	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	0	0	0	0	1	0	1	1	0	0	1	0	0
		0	010	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	0	0	1	0	1	1	1	0	0	0	1	0	0
		0	011	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0	0	1	1	1	0	0	1	0	1	0	0	0	1	0	0	0	1
		0	100																																
		0	101																																
		0	110																																
		0	111																																
		1	000																																
		1	001																																
		1	010																																
		1	011																																
		1	100																																
		1	101																																
		1	110																																
		1	111																																

LDR I Data

Address				Data																															
				IOW	IOR	SPARE	COMPARE	COND JMP EN	Z LOAD	R LOAD	!R OUT	A LOAD	!A OUT	SHL	INV	OR	DEC	SUB	ADD	PASS THRU	!IP OUT	!SYNC IP LOAD	IP CLK EN	OR LOAD	!OR OUT	DROUT LOAD	!DROUT OUT	DRIN LOAD	!DRIN OUT	AR LOAD	!AR OUT	MEMW	MEMR	IR LOAD	IR RESET
OP	ADR	Z	MIP	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
01001	110	0	000	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	1	0	0	1	0	1	0	1	1	0	0	1	0	0
		0	001	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	1	0	1	0	1	1	1	0	0	0	1	0	0
		0	010	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0	0	1	1	1	0	0	1	0	1	0	0	0	1	0	0	0	1
		0	011																																
		0	100																																
		0	101																																
		0	110																																
		0	111																																
		1	000																																
		1	001																																
		1	010																																
		1	011																																
		1	100																																
		1	101																																
		1	110																																
		1	111																																

STA Address

Address				Data																															
				IOW	IOR	SPARE	COMPARE	COND JMP	ZLOAD	RLOAD	!ROUT	ALOAD	!AOUT	SHL	INV	OR	DEC	SUB	ADD	PASSTHRU	!IPOUT	!SYNC IPLOAD	IPCLKEN	ORLOAD	!OROUT	DROUTLOAD	!DROUTOUT	DRINLOAD	!DRINOUT	ARLOAD	!AROUT	MEMW	MEMR	IRLOAD	IRRESET
OP	ADR	Z	MIP	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
01010	000	0	000	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	1	0	0	1	0	1	0	1	1	0	0	1	0	0
		0	001	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	1	0	1	0	1	1	1	0	0	0	1	0	0
		0	010	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	0	1	1	0	1	0	0	0	1	0	0	0	0
		0	011	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	1	1	0	0	0	0	0
		0	100	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	0	0	1	0	0	0	1	0	0	1	0	0	0
		0	101	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	0	0	1	0	0	0	1	0	0	0	0	0	1
		0	110																																
		0	111																																
		1	000																																
		1	001																																
		1	010																																
		1	011																																
		1	100																																
		1	101																																
		1	110																																
		1	111																																

STA M

Address **Data**

OP	ADR	Z	MIP	IOW 31	IOR 30	SPARE 29	COMPARE 28	COND JMP 27	ZLOAD 26	RLOAD 25	!ROUT 24	ALOAD 23	!AOUT 22	SHL 21	INV 20	OR 19	DEC 18	SUB 17	ADD 16	PASSTHRU 15	!IPOUT 14	!SYNC IPLOAD 13	IPCLKEN 12	ORLOAD 11	!OROUT 10	DROUTLOAD 09	!DROUT 08	DRINLOAD 07	!DRINOUT 06	ARLOAD 05	!AROUT 04	MEMW 03	MEMR 02	IRLOAD 01	IRRESET 00
01010	100	0	000	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	1	0	1	1	0	1	0	1	0	1	0	0	0	0
		0	001	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	1	1	0	0	0	0	0
		0	010	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	0	0	1	0	0	0	1	0	0	1	0	0	0
		0	011	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	0	0	1	0	0	0	1	0	0	0	0	0	1
		0	100																																
		0	101																																
		0	110																																
		0	111																																
		1	000																																
		1	001																																
		1	010																																
		1	011																																
		1	100																																
		1	101																																
		1	110																																
		1	111																																

STR Address

Address				Data																															
				IOW	IOR	SPARE	COMPARE	COND JMP EN	ZLOAD	RLOAD	!ROUT	ALOAD	!AOUT	SHL	INV	OR	DEC	SUB	ADD	PASSTHRU	!IPOUT	!SYNC IPLOAD	IPCLKEN	ORLOAD	!OROUT	DROUTLOAD	!DROUT	DRINLOAD	!DRINOUT	ARLOAD	!AROUT			IRLOAD	IRRESET
OP	ADR	Z	MIP	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
01011	000	0	000	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	1	0	0	1	0	1	0	1	1	0	0	1	0	0
		0	001	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	1	0	1	0	1	1	1	0	0	0	1	0	0
		0	010	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	0	1	1	0	1	0	0	0	1	0	0	0	0
		0	011	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	1	1	0	0	0	0	0
		0	100	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	0	0	1	0	0	0	1	0	0	1	0	0	0
		0	101	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	0	0	1	0	0	0	1	0	0	0	0	0	1
		0	110																																
		0	111																																
		1	000																																
		1	001																																
		1	010																																
		1	011																																
		1	100																																
		1	101																																
		1	110																																
		1	111																																

STR M

OP	ADR	Z	MIP	IOW 31	IOR 30	SPARE 29	COMPARE 28	COND JMP EN 27	ZLOAD 26	RLOAD 25	!ROUT 24	ALOAD 23	!AOUT 22	SHL 21	INV 20	OR 19	DEC 18	SUB 17	ADD 16	PASS THRU 15	!IP OUT 14	!SYNC IP LOAD 13	IP CLK EN 12	OR LOAD 11	!OR OUT 10	DROUT LOAD 09	!DROUT OUT 08	DRIN LOAD 07	!DRIN OUT 06	AR LOAD 05	!AR OUT 04	MEMW 03	MEMR 02	IR LOAD 01	IR RESET 00
01011	100	0	000	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	1	0	1	1	0	1	0	1	0	1	0	0	0	0
		0	001	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	1	1	0	0	0	0	0
		0	010	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	0	0	1	0	0	0	1	0	0	1	0	0	0
		0	011	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	0	0	1	0	0	0	1	0	0	0	0	0	1
		0	100																																
		0	101																																
		0	110																																
		0	111																																
		1	000																																
		1	001																																
		1	010																																
		1	011																																
		1	100																																
		1	101																																
		1	110																																
		1	111																																

ADD Address

Address **Data**

OP	ADR	Z	MIP	IOW 31	IOR 30	SPARE 29	COMPARE 28	COND JMP EN 27	ZLOAD 26	RLOAD 25	!ROUT 24	ALOAD 23	!AOUT 22	SHL 21	INV 20	OR 19	DEC 18	SUB 17	ADD 16	PASSTHRU 15	!IPOUT 14	!SYNC IPLOAD 13	IPCLKEN 12	ORLOAD 11	!OROUT 10	DROUTLOAD 09	!DROUT 08	DRINLOAD 07	!DRINOUT 06	ARLOAD 05	!AROUT 04	MEMW 03	MEMR 02	IRLOAD 01	IRRESET 00
01100	000	0	000	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	1	0	0	1	0	1	0	1	1	0	0	1	0	0
		0	001	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	1	0	1	0	1	1	1	0	0	0	1	0	0
		0	010	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	0	1	1	0	1	0	0	0	1	0	0	0	0
		0	011	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	0	0	0	0	1	0	1	1	0	0	1	0	0
		0	100	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	0	0	1	0	1	1	1	0	0	0	1	0	0
		0	101	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	1	0	1	1	0	0	1	0	1	0	0	0	1	0	0	0	1
		0	110																																
		0	111																																
		1	000																																
		1	001																																
		1	010																																
		1	011																																
		1	100																																
		1	101																																
		1	110																																
		1	111																																

ADD A

Address | **Data**

OP	ADR	Z	MIP	IOW 31	IOR 30	SPARE 29	COMPARE 28	COND JMP EN 27	ZLOAD 26	RLOAD 25	!ROUT 24	ALOAD 23	!AOUT 22	SHL 21	INV 20	OR 19	DEC 18	SUB 17	ADD 16	PASSTHRU 15	!IPOUT 14	!SYNC IPLOAD 13	IPCLKEN 12	ORLOAD 11	!OROUT 10	DROUTLOAD 09	!DROUT 08	DRINLOAD 07	!DRINOUT 06	ARLOAD 05	!AROUT 04	MEMW 03	MEMR 02	IRLOAD 01	IRRESET 00
01100	010	0	000	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	0	1	1	0	0	1	0	1	0	1	0	1	0	0	0	1
		0	001																																
		0	010																																
		0	011																																
		0	100																																
		0	101																																
		0	110																																
		0	111																																
		1	000																																
		1	001																																
		1	010																																
		1	011																																
		1	100																																
		1	101																																
		1	110																																
		1	111																																

ADD R

Address **Data**

OP	ADR	Z	MIP	31 IOW	30 IOR	29 SPARE	28 COMPARE	27 COND JMP EN	26 ZLOAD	25 RLOAD	24 !ROUT	23 ALOAD	22 !AOUT	21 SHL	20 INV	19 OR	18 DEC	17 SUB	16 ADD	15 PASS THRU	14 !IPOUT	13 !SYNC IPLOAD	12 IPCLK EN	11 OR LOAD	10 !OROUT	09 DROUTLOAD	08 !DROUT	07 DRIN LOAD	06 !DRIN OUT	05 ARLOAD	04 !AROUT	03 MEMW	02 MEMR	01 IRLOAD	00 IR RESET
01100	011	0	000	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	1	0	1	1	0	0	1	0	1	0	1	0	1	0	0	0	1
		0	001																																
		0	010																																
		0	011																																
		0	100																																
		0	101																																
		0	110																																
		0	111																																
		1	000																																
		1	001																																
		1	010																																
		1	011																																
		1	100																																
		1	101																																
		1	110																																
		1	111																																

ADD M

Address **Data**

OP	ADR	Z	MIP	31 IOW	30 IOR	29 SPARE	28 COMPARE	27 COND JMP EN	26 ZLOAD	25 RLOAD	24 !ROUT	23 ALOAD	22 !AOUT	21 SHL	20 INV	19 OR	18 DEC	17 SUB	16 ADD	15 PASSTHRU	14 !IPOUT	13 !SYNC IPLOAD	12 IPCLKEN	11 ORLOAD	10 !OROUT	09 DROUTLOAD	08 !DROUTOUT	07 DRINLOAD	06 !DRINOUT	05 ARLOAD	04 !AROUT	03 MEMW	02 MEMR	01 IRLOAD	00 IRRESET
01100	100	0	000	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	1	0	1	1	0	1	0	1	0	1	0	0	0	0
		0	001	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	0	0	0	0	1	0	1	1	0	0	1	0	0
		0	010	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	0	0	1	0	1	1	1	0	0	0	1	0	0
		0	011	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	1	0	1	1	0	0	1	0	1	0	0	0	1	0	0	0	1
		0	100																																
		0	101																																
		0	110																																
		0	111																																
		1	000																																
		1	001																																
		1	010																																
		1	011																																
		1	100																																
		1	101																																
		1	110																																
		1	111																																

ADD I Data

Address				Data																															
				IOW	IOR	SPARE	COMPARE	COND JMP EN	ZLOAD	RLOAD	!ROUT	ALOAD	!AOUT	SHL	INV	OR	DEC	SUB	ADD	PASST HRU	!IPOUT	!SYNC IPLOAD	IPCLK EN	ORLOAD	!OROUT	DROUTLOAD	!DROUT	DRINLOAD	!DRINOUT	ARLOAD	!AROUT	MEMW	MEMR	IRLOAD	IRRESET
OP	ADR	Z	MIP	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
01100	110	0	000	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	1	0	0	1	0	1	0	1	1	0	0	1	0	0
		0	001	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	1	0	1	0	1	1	1	0	0	0	1	0	0
		0	010	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	1	0	1	1	0	0	1	0	1	0	0	0	1	0	0	0	1
		0	011																																
		0	100																																
		0	101																																
		0	110																																
		0	111																																
		1	000																																
		1	001																																
		1	010																																
		1	011																																
		1	100																																
		1	101																																
		1	110																																
		1	111																																

SUB Address

Address **Data**

				IOW	IOR	SPARE	COMPARE	COND JMP	ZLOAD	RLOAD	!ROUT	ALOAD	!AOUT	SHL	INV	OR	DEC	SUB	ADD	PASSTHRU	!IPOUT	!SYNC IPLOAD	IPCLKEN	ORLOAD	!OROUT	DROUTLOAD	!DROUT	DRINLOAD	!DRINOUT	ARLOAD	!AROUT	MEMW	MEMR	IRLOAD	IRRESET
OP	ADR	Z	MIP	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
01101	000	0	000	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	1	0	0	1	0	1	0	1	1	0	0	1	0	0
		0	001	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	1	0	1	0	1	1	1	0	0	0	1	0	0
		0	010	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	0	1	1	0	1	0	0	0	1	0	0	0	0
		0	011	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	0	0	0	0	1	0	1	1	0	0	1	0	0
		0	100	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	0	0	1	0	1	1	1	0	0	0	1	0	0
		0	101	0	0	0	0	0	0	0	1	1	1	0	0	0	0	1	0	0	1	1	0	0	1	0	1	0	0	0	1	0	0	0	1
		0	110																																
		0	111																																
		1	000																																
		1	001																																
		1	010																																
		1	011																																
		1	100																																
		1	101																																
		1	110																																
		1	111																																

SUB A

Address | **Data**

OP	ADR	Z	MIP	IOW 31	IOR 30	SPARE 29	COMPARE 28	COND JMP EN 27	ZLOAD 26	RLOAD 25	!ROUT 24	ALOAD 23	!AOUT 22	SHL 21	INV 20	OR 19	DEC 18	SUB 17	ADD 16	PASSTHRU 15	!IPOUT 14	!SYNC IPLOAD 13	IPCLKEN 12	ORLOAD 11	!OROUT 10	DROUTLOAD 09	!DROUT 08	DRINLOAD 07	!DRINOUT 06	ARLOAD 05	!AROUT 04	MEMW 03	MEMR 02	IRLOAD 01	IRRESET 00
01101	010	0	000	0	0	0	0	0	0	0	1	1	0	0	0	0	0	1	0	0	1	1	0	0	1	0	1	0	1	0	1	0	0	0	1
		0	001																																
		0	010																																
		0	011																																
		0	100																																
		0	101																																
		0	110																																
		0	111																																
		1	000																																
		1	001																																
		1	010																																
		1	011																																
		1	100																																
		1	101																																
		1	110																																
		1	111																																

SUB R

Address **Data**

OP	ADR	Z	MIP	IOW 31	IOR 30	SPARE 29	COMPARE 28	COND JMP EN 27	ZLOAD 26	RLOAD 25	!ROUT 24	ALOAD 23	!AOUT 22	SHL 21	INV 20	OR 19	DEC 18	SUB 17	ADD 16	PASSTHRU 15	!IPOUT 14	!SYNC IPLOAD 13	IPCLKEN 12	ORLOAD 11	!OROUT 10	DROUTLOAD 09	!DROUT 08	DRINLOAD 07	!DRINOUT 06	ARLOAD 05	!AROUT 04	MEMW 03	MEMR 02	IRLOAD 01	IRRESET 00
01101	011	0	000	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	0	0	1	1	0	0	1	0	1	0	1	0	1	0	0	0	1
		0	001																																
		0	010																																
		0	011																																
		0	100																																
		0	101																																
		0	110																																
		0	111																																
		1	000																																
		1	001																																
		1	010																																
		1	011																																
		1	100																																
		1	101																																
		1	110																																
		1	111																																

SUB M

Address **Data**

OP	ADR	Z	MIP	IOW	IOR	SPARE	COMPARE	COND JMP	ZLOAD	RLOAD	!ROUT	ALOAD	!AOUT	SHL	INV	OR	DEC	SUB	ADD	PASSTHRU	!IPOUT	!SYNC IPLOAD	IPCLKEN	ORLOAD	!OROUT	DROUTLOAD	!DROUT	DRINLOAD	!DRINOUT	ARLOAD	!AROUT	MEMW	MEMR	IRLOAD	IRRESET
				31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
01101	100	0	000	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	1	0	1	1	0	1	0	1	0	1	0	0	0	0
		0	001	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	0	0	0	0	1	0	1	1	0	0	1	0	0
		0	010	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	0	0	1	0	1	1	1	0	0	0	1	0	0
		0	011	0	0	0	0	0	0	0	1	1	1	0	0	0	0	1	0	0	1	1	0	0	1	0	1	0	0	0	1	0	0	0	1
		0	100																																
		0	101																																
		0	110																																
		0	111																																
		1	000																																
		1	001																																
		1	010																																
		1	011																																
		1	100																																
		1	101																																
		1	110																																
		1	111																																

SUB I Data

Address				Data																															
				IOW	IOR	SPARE	COMPARE	COND JMP EN	ZLOAD	RLOAD	!ROUT	ALOAD	!AOUT	SHL	INV	OR	DEC	SUB	ADD	PASSTHRU	!IPOUT	!SYNC IPLOAD	IPCLKEN	ORLOAD	!OROUT	DROUTLOAD	!DROUT	DRINLOAD	!DRINOUT	ARLOAD	!AROUT	MEMW	MEMR	IRLOAD	IRRESET
OP	ADR	Z	MIP	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
01101	110	0	000	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	1	0	0	1	0	1	0	1	1	0	0	1	0	0
		0	001	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	1	0	1	0	1	1	1	0	0	0	1	0	0
		0	010	0	0	0	0	0	0	0	1	1	1	0	0	0	0	1	0	0	1	1	0	0	1	0	1	0	0	0	1	0	0	0	1
		0	011																																
		0	100																																
		0	101																																
		0	110																																
		0	111																																
		1	000																																
		1	001																																
		1	010																																
		1	011																																
		1	100																																
		1	101																																
		1	110																																
		1	111																																

DEC Address

Address				Data																													
				IOW	IOR	SPARE	COMPARE	COND JMP	ZLOAD	RLOAD	!ROUT	ALOAD	!AOUT	SHL	INV	OR	DEC	SUB	ADD	PASSTHRU	!IPOUT	!SYNC IPLOAD	IPCLKEN	ORLOAD	!OROUT	DROUTLOAD	!DROUTOUT	DRINLOAD	!DRINOUT	ARLOAD	!AROUT	MEMW	MEMR	IRLOAD	IRRESET
OP	ADR	Z	MIP	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
01110	000	0	000	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	1	0	0	1	0	1	0	1	1	0	0	1	0	0
		0	001	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	1	0	1	0	1	1	1	0	0	0	1	0	0
		0	010	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	0	1	1	0	1	0	0	0	1	0	0	0	0
		0	011	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	0	0	0	0	1	0	1	1	0	0	1	0	0
		0	100	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	0	0	1	0	1	1	1	0	0	0	1	0	0
		0	101	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	1	1	1	0	0	1	0	1	0	0	0	1	0	0	0
		0	110	0	0	0	0	0	0	0	1	0	1	0	0	0	1	0	0	0	1	1	0	0	1	0	1	0	1	0	1	0	0	0	0
		0	111	0	0	0	0	0	0	0	1	1	1	0	0	0	1	0	0	0	1	1	0	0	1	0	1	0	1	0	1	0	0	0	1
		1	000																																
		1	001																																
		1	010																																
		1	011																																
		1	100																																
		1	101																																
		1	110																																
		1	111																																

DEC A

Address **Data**

OP	ADR	Z	MIP	IOW 31	IOR 30	SPARE 29	COMPARE 28	COND JMP EN 27	ZLOAD 26	RLOAD 25	!ROUT 24	ALOAD 23	!AOUT 22	SHL 21	INV 20	OR 19	DEC 18	SUB 18	ADD 16	PASSTHRU 15	!IPOUT 14	!SYNC IPLOAD 13	IPCLKEN 12	ORLOAD 11	!OROUT 10	DROUTLOAD 09	!DROUT 08	DRINLOAD 07	!DRINOUT 06	ARLOAD 05	!AROUT 04	MEMW 03	MEMR 02	IRLOAD 01	IRRESET 00
01110	010	0	000	0	0	0	0	0	0	0	1	1	1	0	0	0	1	0	0	0	1	1	0	0	1	0	1	0	1	0	1	0	0	0	1
		0	001																																
		0	010																																
		0	011																																
		0	100																																
		0	101																																
		0	110																																
		0	111																																
		1	000																																
		1	001																																
		1	010																																
		1	011																																
		1	100																																
		1	101																																
		1	110																																
		1	111																																

DEC R

Address **Data**

OP	ADR	Z	MIP	IOW 31	IOR 30	SPARE 29	COMPARE 28	COND JMP 27	ZLOAD 26	RLOAD 25	!ROUT 24	ALOAD 23	!AOUT 22	SHL 21	INV 20	OR 19	DEC 18	SUB 18	ADD 16	PASSTHRU 15	!IPOUT 14	!SYNC IPLOAD 13	IPCLKEN 12	ORLOAD 11	!OROUT 10	DROUTLOAD 09	!DROUT 08	DRINLOAD 07	!DRINOUT 06	ARLOAD 05	!AROUT 04	MEMW 03	MEMR 02	IRLOAD 01	IRRESET 00
01110	011	0	000	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	1	1	0	0	1	0	1	0	1	0	1	0	0	0	0
		0	001	0	0	0	0	0	0	0	1	0	1	0	0	0	1	0	0	0	1	1	0	0	1	0	1	0	1	0	1	0	0	0	0
		0	010	0	0	0	0	0	0	0	1	1	1	0	0	0	1	0	0	0	1	1	0	0	1	0	1	0	1	0	1	0	0	0	0
		0	011	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0	1	0	1	0	1	0	0	0	1
		0	100																																
		0	101																																
		0	110																																
		0	111																																
		1	000																																
		1	001																																
		1	010																																
		1	011																																
		1	100																																
		1	101																																
		1	110																																
		1	111																																

DEC M

Address **Data**

OP	ADR	Z	MIP	31 IOW	30 IOR	29 SPARE	28 COMPARE	27 COND JMP EN	26 ZLOAD	25 RLOAD	24 !ROUT	23 ALOAD	22 !AOUT	21 SHL	20 INV	19 OR	18 DEC	17 SUB	16 ADD	15 PASS THRU	14 !IPOUT	13 !SYNC IPLOAD	12 IPCLK EN	11 ORLOAD	10 !OROUT	09 DROUT LOAD	08 !DROUT OUT	07 DRIN LOAD	06 !DRIN OUT	05 ARLOAD	04 !AROUT	03 MEMW	02 MEMR	01 IRLOAD	00 IR RESET
01110	100	0	000	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	1	0	1	1	0	1	0	1	0	1	0	0	0	0
		0	001	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	0	0	0	0	1	0	1	1	0	0	1	0	0
		0	010	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	0	0	1	0	1	1	1	0	0	0	1	0	0
		0	011	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	1	1	1	0	0	1	0	1	0	0	0	1	0	0	0
		0	100	0	0	0	0	0	0	0	1	0	1	0	0	0	1	0	0	0	1	1	0	0	1	0	1	0	1	0	1	0	0	0	0
		0	101	0	0	0	0	0	0	0	1	1	1	0	0	0	1	0	0	0	1	1	0	0	1	0	1	0	1	0	1	0	0	0	1
		0	110																																
		0	111																																
		1	000																																
		1	001																																
		1	010																																
		1	011																																
		1	100																																
		1	101																																
		1	110																																
		1	111																																

DEC I Data

Address | **Data**

OP	ADR	Z	MIP	31 IOW	30 IOR	29 SPARE	28 COMPARE	27 COND JMP EN	26 ZLOAD	25 RLOAD	24 !ROUT	23 ALOAD	22 !AOUT	21 SHL	20 INV	19 OR	18 DEC	17 SUB	16 ADD	15 PASS THRU	14 !IPOUT	13 !SYNC IPLOAD	12 IPCLK EN	11 ORLOAD	10 !OROUT	09 DROUTLOAD	08 !DROUTOUT	07 DRINLOAD	06 !DRINOUT	05 ARLOAD	04 !AROUT	03 MEMW	02 MEMR	01 IRLOAD	00 IR RESET
01110	110	0	000	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	1	0	0	1	0	1	0	1	1	0	0	1	0	0
		0	001	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	1	0	1	0	1	1	1	0	0	0	1	0	0
		0	010	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	1	1	1	0	0	1	0	1	0	0	0	1	0	0	0	0
		0	011	0	0	0	0	0	0	0	1	0	1	0	0	0	1	0	0	0	1	1	0	0	1	0	1	0	1	0	1	0	0	0	0
		0	100	0	0	0	0	0	0	0	1	1	1	0	0	0	1	0	0	0	1	1	0	0	1	0	1	0	1	0	1	0	0	0	1
		0	101																																
		0	110																																
		0	111																																
		1	000																																
		1	001																																
		1	010																																
		1	011																																
		1	100																																
		1	101																																
		1	110																																
		1	111																																

OR Address

Address **Data**

				IOW	IOR	SPARE	COMPARE	COND JMP	ZLOAD	RLOAD	!ROUT	ALOAD	!AOUT	SHL	INV	OR	DEC	SUB	ADD	PASSTHRU	!IPOUT	!SYNC IPLOAD	IPCLKEN	ORLOAD	!OROUT	DROUTLOAD	!DROUT	DRINLOAD	!DRINOUT	ARLOAD	!AROUT	MEMW	MEMR	IRLOAD	IRRESET
OP	ADR	Z	MIP	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
10000	000	0	000	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	1	0	0	1	0	1	0	1	1	0	0	1	0	0
		0	001	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	1	0	1	0	1	1	1	0	0	0	1	0	0
		0	010	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	0	1	1	0	1	0	0	0	1	0	0	0	0
		0	011	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	0	0	0	0	1	0	1	1	0	0	1	0	0
		0	100	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	0	0	1	0	1	1	1	0	0	0	1	0	0
		0	101	0	0	0	0	0	0	0	1	1	1	0	0	1	0	0	0	0	1	1	0	0	1	0	1	0	0	0	1	0	0	0	1
		0	110																																
		0	111																																
		1	000																																
		1	001																																
		1	010																																
		1	011																																
		1	100																																
		1	101																																
		1	110																																
		1	111																																

OR A

Address **Data**

OP	ADR	Z	MIP	IOW 31	IOR 30	SPARE 29	COMPARE 28	COND JMP EN 27	ZLOAD 26	RLOAD 25	!ROUT 24	ALOAD 23	!AOUT 22	SHL 21	INV 20	OR 19	DEC 18	SUB 17	ADD 16	PASSTHRU 15	!IPOUT 14	!SYNC IPLOAD 13	IPCLKEN 12	ORLOAD 11	!OROUT 10	DROUTLOAD 09	!DROUT 08	DRINLOAD 07	!DRINOUT 06	ARLOAD 05	!AROUT 04	MEMW 03	MEMR 02	IRLOAD 01	IRRESET 00
10000	010	0	000	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0	0	1	1	0	0	1	0	1	0	1	0	1	0	0	0	1
		0	001																																
		0	010																																
		0	011																																
		0	100																																
		0	101																																
		0	110																																
		0	111																																
		1	000																																
		1	001																																
		1	010																																
		1	011																																
		1	100																																
		1	101																																
		1	110																																
		1	111																																

OR R

Address **Data**

OP	ADR	Z	MIP	IOW 31	IOR 30	SPARE 29	COMPARE 28	COND JMP EN 27	ZLOAD 26	RLOAD 25	!ROUT 24	ALOAD 23	!AOUT 22	SHL 21	INV 20	OR 19	DEC 18	SUB 17	ADD 16	PASSTHRU 15	!IPOUT 14	!SYNC IPLOAD 13	IPCLKEN 12	ORLOAD 11	!OROUT 10	DROUTLOAD 09	!DROUT 08	DRINLOAD 07	!DRINOUT 06	ARLOAD 05	!AROUT 04	MEMW 03	MEMR 02	IRLOAD 01	IRRESET 00
10000	011	0	000	0	0	0	0	0	0	0	0	1	1	0	0	1	0	0	0	0	1	1	0	0	1	0	1	0	1	0	1	0	0	0	1
		0	001																																
		0	010																																
		0	011																																
		0	100																																
		0	101																																
		0	110																																
		0	111																																
		1	000																																
		1	001																																
		1	010																																
		1	011																																
		1	100																																
		1	101																																
		1	110																																
		1	111																																

OR M

Address **Data**

OP	ADR	Z	MIP	31 IOW	30 IOR	29 SPARE	28 COMPARE	27 COND JMP EN	26 ZLOAD	25 RLOAD	24 !ROUT	23 ALOAD	22 !AOUT	21 SHL	20 INV	19 OR	18 DEC	17 SUB	16 ADD	15 PASSTHRU	14 !IPOUT	13 !SYNC IPLOAD	12 IPCLKEN	11 ORLOAD	10 !OROUT	09 DROUTLOAD	08 !DROUTOUT	07 DRINLOAD	06 !DRINOUT	05 ARLOAD	04 !AROUT	03 MEMW	02 MEMR	01 IRLOAD	00 IRRESET
10000	100	0	000	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	1	0	1	1	0	1	0	1	0	1	0	0	0	0
		0	001	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	0	0	0	0	1	0	1	1	0	0	1	0	0
		0	010	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	0	0	1	0	1	1	1	0	0	0	1	0	0
		0	011	0	0	0	0	0	0	0	1	1	1	0	0	1	0	0	0	0	1	1	0	0	1	0	1	0	0	0	1	0	0	0	1
		0	100																																
		0	101																																
		0	110																																
		0	111																																
		1	000																																
		1	001																																
		1	010																																
		1	011																																
		1	100																																
		1	101																																
		1	110																																
		1	111																																

OR I Data

Address				Data																															
				IOW	IOR	SPARE	COMPARE	COND JMP EN	ZLOAD	RLOAD	!ROUT	ALOAD	!AOUT	SHL	INV	OR	DEC	SUB	ADD	PASST HRU	!IPOUT	!SYNC IPLOAD	IPCLK EN	ORLOAD	!OROUT	DROUTLOAD	!DROUT	DRINLOAD	!DRINOUT	ARLOAD	!AROUT	MEMW	MEMR	IRLOAD	IRRESET
OP	ADR	Z	MIP	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
10000	110	0	000	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	1	0	0	1	0	1	0	1	1	0	0	1	0	0
		0	001	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	1	0	1	0	1	1	1	0	0	0	1	0	0
		0	010	0	0	0	0	0	0	0	1	1	1	0	0	1	0	0	0	0	1	1	0	0	1	0	1	0	0	0	1	0	0	0	1
		0	011																																
		0	100																																
		0	101																																
		0	110																																
		0	111																																
		1	000																																
		1	001																																
		1	010																																
		1	011																																
		1	100																																
		1	101																																
		1	110																																
		1	111																																

INV Address

Address **Data**

OP	ADR	Z	MIP	31 IOW	30 IOR	29 SPARE	28 COMPARE	27 COND JMP EN	26 ZLOAD	25 RLOAD	24 !ROUT	23 ALOAD	22 !AOUT	21 SHL	20 INV	19 OR	18 DEC	17 SUB	16 ADD	15 PASSTHRU	14 !IPOUT	13 !SYNC IPLOAD	12 IPCLKEN	11 ORLOAD	10 !OROUT	09 DROUTLOAD	08 !DROUTOUT	07 DRINLOAD	06 !DRINOUT	05 ARLOAD	04 !AROUT	03 MEMW	02 MEMR	01 IRLOAD	00 IRRESET
10001	000	0	000	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	0	0	1	0	1	0	1	1	0	0	1	0	0
		0	001	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	1	0	1	0	1	1	1	0	0	0	1	0	0
		0	010	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	0	1	1	0	1	0	0	0	1	0	0	0	0
		0	011	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	0	0	0	0	1	0	1	1	0	0	1	0	0
		0	100	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	0	0	1	0	1	1	1	0	0	0	1	0	0
		0	101	0	0	0	0	0	0	0	1	1	1	0	1	0	0	0	0	0	1	1	0	0	1	0	1	0	0	0	1	0	0	0	1
		0	110																																
		0	111																																
		1	000																																
		1	001																																
		1	010																																
		1	011																																
		1	100																																
		1	101																																
		1	110																																
		1	111																																

INV A

Address **Data**

OP	ADR	Z	MIP	31 IOW	30 IOR	29 SPARE	28 COMPARE	27 COND JMP EN	26 ZLOAD	25 RLOAD	24 !ROUT	23 ALOAD	22 !AOUT	21 SHL	20 INV	19 OR	18 DEC	17 SUB	16 ADD	15 PASSTHRU	14 !IPOUT	13 !SYNC IPLOAD	12 IPCLKEN	11 ORLOAD	10 !OROUT	09 DROUTLOAD	08 !DROUTOUT	07 DRINLOAD	06 !DRINOUT	05 ARLOAD	04 !AROUT	03 MEMW	02 MEMR	01 IRLOAD	00 IRRESET
10001	010	0	000	0	0	0	0	0	0	0	1	1	0	0	1	0	0	0	0	0	1	1	0	0	1	0	1	0	1	0	1	0	0	0	1
		0	001																																
		0	010																																
		0	011																																
		0	100																																
		0	101																																
		0	110																																
		0	111																																
		1	000																																
		1	001																																
		1	010																																
		1	011																																
		1	100																																
		1	101																																
		1	110																																
		1	111																																

INV R

Address **Data**

OP	ADR	Z	MIP	31 IOW	30 IOR	29 SPARE	28 COMPARE	27 COND JMP EN	26 ZLOAD	25 RLOAD	24 !ROUT	23 ALOAD	22 !AOUT	21 SHL	20 INV	19 OR	18 DEC	17 SUB	16 ADD	15 PASSTHRU	14 !IPOUT	13 !SYNC IPLOAD	12 IPCLKEN	11 ORLOAD	10 !OROUT	09 DROUTLOAD	08 !DROUTOUT	07 DRINLOAD	06 !DRINOUT	05 ARLOAD	04 !AROUT	03 MEMW	02 MEMR	01 IRLOAD	00 IRRESET
10001	011	0	000	0	0	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0	1	1	0	0	1	0	1	0	1	0	1	0	0	0	0
		0	001	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0	1	0	1	0	1	0	0	0	1
		0	010																																
		0	011																																
		0	100																																
		0	101																																
		0	110																																
		0	111																																
		1	000																																
		1	001																																
		1	010																																
		1	011																																
		1	100																																
		1	101																																
		1	110																																
		1	111																																

INV M

Address **Data**

OP	ADR	Z	MIP	31 IOW	30 IOR	29 SPARE	28 COMPARE	27 COND JMP EN	26 ZLOAD	25 RLOAD	24 !ROUT	23 ALOAD	22 !AOUT	21 SHL	20 INV	19 OR	18 DEC	17 SUB	16 ADD	15 PASS THRU	14 !IPOUT	13 !SYNC IPLOAD	12 IPCLK EN	11 ORLOAD	10 !OROUT	09 DROUTLOAD	08 !DROUTOUT	07 DRINLOAD	06 !DRINOUT	05 ARLOAD	04 !AROUT	03 MEMW	02 MEMR	01 IRLOAD	00 IRRESET
10001	100	0	000	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	1	0	1	1	0	1	0	1	0	1	0	0	0	0
		0	001	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	0	0	0	0	1	0	1	1	0	0	1	0	0
		0	010	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	0	0	1	0	1	1	1	0	0	0	1	0	0
		0	011	0	0	0	0	0	0	0	1	1	1	0	1	0	0	0	0	0	1	1	0	0	1	0	1	0	0	0	1	0	0	0	1
		0	100																																
		0	101																																
		0	110																																
		0	111																																
		1	000																																
		1	001																																
		1	010																																
		1	011																																
		1	100																																
		1	101																																
		1	110																																
		1	111																																

INV I Data

Address **Data**

OP	ADR	Z	MIP	31 IOW	30 IOR	29 SPARE	28 COMPARE	27 COND JMP EN	26 ZLOAD	25 RLOAD	24 !ROUT	23 ALOAD	22 !AOUT	21 SHL	20 INV	19 OR	18 DEC	17 SUB	16 ADD	15 PASSTHRU	14 !IPOUT	13 !SYNC IPLOAD	12 IPCLKEN	11 ORLOAD	10 !OROUT	09 DROUTLOAD	08 !DROUTOUT	07 DRINLOAD	06 !DRINOUT	05 ARLOAD	04 !AROUT	03 MEMW	02 MEMR	01 IRLOAD	00 IRRESET
10001	110	0	000	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	1	0	0	1	0	1	0	1	1	0	0	1	0	0
		0	001	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	1	0	1	0	1	1	1	0	0	0	1	0	0
		0	010	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	0	0	1	0	1	0	0	0	1	0	0	0	1
		0	011																																
		0	100																																
		0	101																																
		0	110																																
		0	111																																
		1	000																																
		1	001																																
		1	010																																
		1	011																																
		1	100																																
		1	101																																
		1	110																																
		1	111																																

SHL Address

Address				Data																													
				IOW	IOR	SPARE	COMPARE	COND JMP	ZLOAD	RLOAD	!ROUT	ALOAD	!AOUT	SHL	INV	OR	DEC	SUB	ADD	PASSTHRU	!IPOUT	!SYNC IPLOAD	IPCLKEN	ORLOAD	!OROUT	DROUTLOAD	!DROUTOUT	DRINLOAD	!DRINOUT	ARLOAD	!AROUT	MEMW	MEMR	IRLOAD	IRRESET
OP	ADR	Z	MIP	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
10010	000	0	000	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	1	0	0	1	0	1	0	1	1	0	0	1	0	0
		0	001	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	1	0	1	0	1	1	1	0	0	0	1	0	0
		0	010	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	0	1	1	0	1	0	0	0	1	0	0	0	0
		0	011	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	0	0	0	0	1	0	1	1	0	0	1	0	0
		0	100	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	0	0	1	0	1	1	1	0	0	0	1	0	0
		0	101	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	1	1	1	0	0	1	0	1	0	0	0	1	0	0	0
		0	110	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	1	1	0	0	1	0	1	0	1	0	1	0	0	0	0
		0	111	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	1	1	0	0	1	0	1	0	1	0	1	0	0	0	1
		1	000																																
		1	001																																
		1	010																																
		1	011																																
		1	100																																
		1	101																																
		1	110																																
		1	111																																

SHL A

Address				Data																															
				IOW	IOR	SPARE	COMPARE	COND JMP EN	ZLOAD	RLOAD	!ROUT	ALOAD	!AOUT	SHL	INV	OR	DEC	SUB	ADD	PASST HRU	!IPOUT	!SYNC IPLOAD	IPCLK EN	ORLOAD	!OROUT	DROUTLOAD	!DROUT	DRINLOAD	!DRINOUT	ARLOAD	!AROUT	MEMW	MEMR	IRLOAD	IRRESET
OP	ADR	Z	MIP	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
10010	010	0	000	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	1	1	0	0	1	0	1	0	1	0	1	0	0	0	1
		0	001																																
		0	010																																
		0	011																																
		0	100																																
		0	101																																
		0	110																																
		0	111																																
		1	000																																
		1	001																																
		1	010																																
		1	011																																
		1	100																																
		1	101																																
		1	110																																
		1	111																																

SHL R

Address **Data**

OP	ADR	Z	MIP	31 IOW	30 IOR	29 SPARE	28 COMPARE	27 COND JMP EN	26 ZLOAD	25 RLOAD	24 !ROUT	23 ALOAD	22 !AOUT	21 SHL	20 INV	19 OR	18 DEC	17 SUB	16 ADD	15 PASSTHRU	14 !IPOUT	13 !SYNC IPLOAD	12 IPCLKEN	11 ORLOAD	10 !OROUT	09 DROUTLOAD	08 !DROUTOUT	07 DRINLOAD	06 !DRINOUT	05 ARLOAD	04 !AROUT	03 MEMW	02 MEMR	01 IRLOAD	00 IRRESET
10010	011	0	000	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	1	1	0	0	1	0	1	0	1	0	1	0	0	0	0
		0	001	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	1	1	0	0	1	0	1	0	1	0	1	0	0	0	0
		0	010	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	1	1	0	0	1	0	1	0	1	0	1	0	0	0	0
		0	011	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0	1	0	1	0	1	0	0	0	1
		0	100																																
		0	101																																
		0	110																																
		0	111																																
		1	000																																
		1	001																																
		1	010																																
		1	011																																
		1	100																																
		1	101																																
		1	110																																
		1	111																																

SHL M

Address				Data																															
				IOW	IOR	SPARE	COMPARE	COND JMP	ZLOAD	RLOAD	!ROUT	ALOAD	!AOUT	SHL	INV	OR	DEC	SUB	ADD	PASSTHRU	!IPOUT	!SYNC IPLOAD	IPCLKEN	ORLOAD	!OROUT	DROUTLOAD	!DROUT	DRINLOAD	!DRINOUT	ARLOAD	!AROUT	MEMW	MEMR	IRLOAD	IRRESET
OP	ADR	Z	MIP	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
10010	100	0	000	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	1	0	1	1	0	1	0	1	0	1	0	0	0	0
		0	001	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	0	0	0	0	1	0	1	1	0	0	1	0	0
		0	010	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	0	0	1	0	1	1	1	0	0	0	1	0	0
		0	011	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	1	1	1	0	0	1	0	1	0	0	0	1	0	0	0
		0	100	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	1	1	0	0	1	0	1	0	1	0	1	0	0	0	0
		0	101	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	1	1	0	0	1	0	1	0	1	0	1	0	0	0	1
		0	110																																
		0	111																																
		1	000																																
		1	001																																
		1	010																																
		1	011																																
		1	100																																
		1	101																																
		1	110																																
		1	111																																

SHL I Data

Address				Data																													
				IOW	IOR	SPARE	COMPARE	COND JMP	ZLOAD	RLOAD	!ROUT	ALOAD	!AOUT	SHL	INV	OR	DEC	SUB	ADD	PASSTHRU	!IPOUT	!SYNC IPLOAD	IPCLKEN	OR LOAD	!OROUT	DROUTLOAD	!DROUT	DRIN LOAD	!DRINOUT	ARLOAD	!AROUT	MEMW	MEMR	IRLOAD	IRRESET
OP	ADR	Z	MIP	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
10010	110	0	000	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	1	0	0	1	0	1	0	1	1	0	0	1	0	0
		0	001	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	1	0	1	0	1	1	1	0	0	0	1	0	0
		0	010	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	1	1	1	0	0	1	0	1	0	0	0	1	0	0	0	0
		0	011	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	1	1	0	0	1	0	1	0	1	0	1	0	0	0	0
		0	100	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	1	1	0	0	1	0	1	0	1	0	1	0	0	0	1
		0	101																																
		0	110																																
		0	111																																
		1	000																																
		1	001																																
		1	010																																
		1	011																																
		1	100																																
		1	101																																
		1	110																																
		1	111																																

Each instruction has an 8-bit opcode. The eight bits are divided into two fields: 1) the operation; and 2) the addressing mode.

All instructions that use register addressing or indirect addressing require only one byte. A second byte is required for the other two addressing modes. The second byte of a direct instruction contains the 8-bit address, and the second byte of an immediate instruction specifies the immediate data. See Figure 7.2.1.

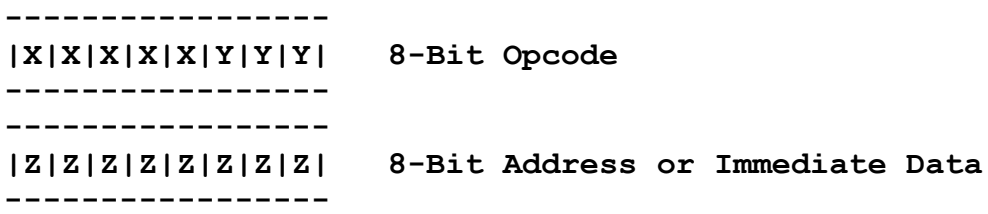


Figure 7.1.1: Instruction Format

The operation is encoded in the 5-bit field labeled 'XXXXX':

5-Bit Operation Code	Instruction Type	Category
00001	IN	Input/Output
00010	OUT	Input/Output
00100	JMP	Program Control
00101	JNZ	Program Control
00110	JZ	Program Control
00111	CMP	Program Control
01000	LDA	Data Transfer
01001	LDR	Data Transfer
01010	STA	Data Transfer
01011	STR	Data Transfer
01100	ADD	Arithmetic
01101	SUB	Arithmetic
01110	DEC	Arithmetic
10000	OR	Logical
10001	INV	Logical
10010	SHL	Logical

The addressing mode is encoded in the 3-bit field labeled 'YYY':

3-Bit Code	Addressing Mode	Data Location
000	Direct	Memory or Port Address in Byte 2
001	Not Used	---
010	Register	A Register
011	Register	R Register
100	Indirect	Memory or Port Address in R Register
101	Not Used	---
110	Immediate	Byte 2 of Instruction
111	Not Used	—

Complete Instruction List

Hexadecimal Object Code	P8 Assembly Source Code
08, address	IN address
0C	IN P
10, address	OUT address
14	OUT P
20, address	JMP address
23	JMP R
28, address	JNZ address
2B	JNZ R
30, address	JZ address
33	JZ R
38, address	CMP address
3A	CMP A
3B	CMP R
3C	CMP M
3E, data	CMP I data
40, address	LDA address
42	LDA A
43	LDA R
44	LDA M
46, data	LDA I data
48, address	LDR address
4A	LDR A
4B	LDR R
4C	LDR M
4E, data	LDR I data
50, address	STA address
54	STA M
58, address	STR address
5C	STR M

Hexadecimal Object Code	P8 Assembly Source Code
60, address	ADD address
62	ADD A
63	ADD R
64	ADD M
66, data	ADD I data
68, address	SUB address
6A	SUB A
6B	SUB R
6C	SUB M
6E, data	SUB I data
70, address	DEC address
72	DEC A
73	DEC R
74	DEC M
76, data	DEC I data
80, address	OR address
82	OR A
83	OR R
84	OR M
86, data	OR I data
88, address	INV address
8A	INV A
8B	INV R
8C	INV M
8E, data	INV I, data
90, address	SHL address
92	SHL A
93	SHL R
94	SHL M
96, data	SHL, data
00	FETCH

FETCH — **Opcode Fetch**

NOTE: **FETCH is not generally used explicitly by programmers. Instead, it is automatically invoked by the CPU to read the next instruction that is to be executed.**

It is described here because it is part of every P8 CPU instruction. The first three clock cycles of any instruction are used by FETCH.

Operation: **IR <-- MEM(IP); IP <-- IP + 1**

Encoding: **00000000 (00h)**

Clock Cycles: **3**

Description: **The contents of the memory address pointed to by the Instruction Pointer (IP) are transferred to the Instruction Register (IR). IP increments.**

Example:

Preconditions:	**Address 06h =**	**54h**
	IR =	**70h**
	IP =	**06h**
Instruction:	**FETCH**	
Postconditions:	**IR =**	**54h**
	IP =	**07h**

Micro-Instructions: **AR <-- IP; Memory Address <-- AR; Assert MEMR**

DR <-- Memory Data; IP <-- IP + 1;
Memory Address <-- AR; Assert MEMR

IR <-- DR

ADD address **Add Direct To A Register**

Addressing Mode: **Direct**

Operation: **A <-- A + MEM(address); IP <-- IP + 2**

Encoding: **Byte 1: 01100 000 (60h)**
Byte 2: 8-bit address

Clock Cycles: **9**

Description: **The contents of the memory address in byte 2 are added to the contents of the A Register. The result is stored in the A Register. IP increments 2 times.**

Example:

Preconditions:	**Address 10h =**	**04h**
	A Register =	**06h**
	IP =	**00h**
Instruction:	**ADD 10h**	
Postconditions:	**A Register =**	**0Ah**
	IP =	**02h**

Micro-Instructions: **Op Code Fetch (3 Clock Cycles)**

AR <-- IP; Memory Address <-- AR; Assert MEMR

DR <-- Memory Data; IP <-- IP + 1; Memory Address <-- AR; Assert MEMR

OR <-- DR

AR <-- OR; Memory Address <-- AR; Assert MEMR

DR <-- Memory Data; Memory Address <-- AR; Assert MEMR

ALU(A) <-- A; ALU(B) <-- DR;
ALU(F) <-- ALU(A) + ALU(B);
A <-- ALU(F)

ADD A — **Add A Register To A Register**

Addressing Mode: Register

Operation: A <-- A + A; IP <-- IP + 1

Encoding: 01100 010 (62h)

Clock Cycles: 4

Description: The contents of the A Register are added to the contents of the A Register. The result is stored in the A Register. IP increments.

Example:

Preconditions:	**A Register =**	**06h**
	IP =	**00h**
Instruction:	**ADD A**	
Postconditions:	**A Register =**	**0Ch**
	IP =	**01h**

Micro-Instructions: Op Code Fetch (3 Clock Cycles)

ALU(A) <-- A; ALU(B) <-- A;
ALU(F) <-- ALU(A) + ALU(B);
A <-- ALU(F)

ADD R — **Add R Register To A Register**

Addressing Mode: Register

Operation: A <-- A + R; IP <-- IP + 1

Encoding: 01100 011 (63h)

Clock Cycles: 4

Description: The contents of the R Register are added to the contents of the A Register. The result is stored in the A Register. IP increments.

Example:

Preconditions:	**R Register =**	**04h**
	A Register =	**06h**
	IP =	**00h**
Instruction:	**ADD R**	
Postconditions:	**A Register =**	**0Ah**
	IP =	**01h**

Micro-Instructions: Op Code Fetch (3 Clock Cycles)

ALU(A) <-- A; ALU(B) <-- R;
ALU(F) <-- ALU(A) + ALU(B);
A <-- ALU(F)

ADD M **Add Indirect To A Register**

Addressing Mode: **Indirect**

Operation: **A <-- A + MEM(R); IP <-- IP + 1**

Encoding: **01100 100 (64h)**

Clock Cycles: **7**

Description: **The contents of the memory address pointed to by the R Register are added to the contents of the A Register. The result is stored in the A Register. IP increments.**

Example:

Preconditions:	**Address 10h =**	**04h**
	R Register =	**10h**
	A Register =	**06h**
	IP =	**00h**
Instruction:	**ADD M**	
Postconditions:	**A Register =**	**0Ah**
	IP =	**01h**

Micro-Instructions: **Op Code Fetch (3 Clock Cycles)**

OR <-- R

AR <-- OR; Memory Address <-- AR; Assert MEMR

DR <-- Memory Data; Memory Address <-- AR; Assert MEMR

ALU(A) <-- A; ALU(B) <-- DR;
ALU(F) <-- ALU(A) + ALU(B);
A <-- ALU(F)

ADD I data **Add Immediate To A Register**

Addressing Mode: **Immediate**

Operation: **A <-- A + data; IP <-- IP + 2**

Encoding: **Byte 1: 01100 110 (66h)**
Byte 2: 8-bit data

Clock Cycles: **6**

Description: **The data in byte 2 are added to the contents of the A Register. The result is stored in the A Register. IP increments 2 times.**

Example:

Preconditions:	**A Register =**	**06h**
	IP =	**00h**
Instruction:	**ADD 10h**	
Postconditions:	**A Register =**	**16h**
	IP =	**02h**

Micro-Instructions: **Op Code Fetch (3 Clock Cycles)**

AR <-- IP; Memory Address <-- AR; Assert MEMR

DR <-- Memory Data; IP <-- IP + 1; Memory Address <-- AR; Assert MEMR

ALU(A) <-- A; ALU(B) <-- DR;
ALU(F) <-- ALU(A) + ALU(B);
A <-- ALU(F)

CMP address **Compare Direct With A Register**

Addressing Mode: **Direct**

Operation: **if A - MEM(address) = 0: Z <-- 1; IP <-- IP + 2**

Encoding: **Byte 1: 00111 000 (38h)**
Byte 2: 8-bit address

Clock Cycles: **9**

Description: **The contents of the memory address in byte 2 are compared (by subtraction) to the A Register. If they are equal, the Zero Flag is set, otherwise it is reset. IP increments 2 times.**

Example:

Preconditions:	**Address 10h =**	**06h**
	A Register =	**06h**
	IP =	**00h**
	Z =	**don't care**
Instruction:	**CMP 10h**	
Postconditions:	**A Register =**	**06h**
	IP =	**02h**
	Z =	**1h**

Micro-Instructions: **Op Code Fetch (3 Clock Cycles)**

AR <-- IP; Memory Address <-- AR; Assert MEMR

DR <-- Memory Data; IP <-- IP + 1; Memory Address <-- AR; Assert MEMR

OR <-- DR

AR <-- OR; Memory Address <-- AR; Assert MEMR

DR <-- Memory Data; Memory Address <-- AR; Assert MEMR

ALU(A) <-- A; ALU(B) <-- DR; ALU(F) <-- ALU(A) - ALU(B);
Z <-- Zero Condition Bit

CMP A **Compare A Register With A Register**

Addressing Mode: **Register**

Operation: **Z <-- 1; IP <-- IP + 1**

Encoding: **00111 010 (3Ah)**

Clock Cycles: **4**

Description: **The contents of the A Register are subtracted from the contents of the A Register. The result is always zero, which sets the Zero Flag. The result is discarded. IP increments.**

Example:

Preconditions:	**A Register =**	**06h**
	IP =	**00h**
	Z =	**don't care**
Instruction:	**CMP A**	
Postconditions:	**A Register =**	**06h**
	IP =	**01h**
	Z =	**1h**

Micro-Instructions: **Op Code Fetch (3 Clock Cycles)**

ALU(A) <-- A; ALU(B) <-- A;
ALU(F) <-- ALU(A) - ALU(B);
Z <-- 1

CMP R **Compare R Register With A Register**

Addressing Mode: **Register**

Operation: **if A - R = 0: Z <-- 1; IP <-- IP + 1**

Encoding: **00111 011 (3Bh)**

Clock Cycles: **4**

Description: **The contents of the R Register are subtracted from the contents of the A Register. If the result is zero, the Zero Flag is set, otherwise it is reset. The result is discarded. IP increments.**

Example:

Preconditions:	**R Register =**	**06h**
	A Register =	**06h**
	IP =	**00h**
	Z =	**don't care**
Instruction:	**CMP R**	
Postconditions:	**R Register =**	**06h**
	A Register =	**06h**
	IP =	**01h**
	Z =	**1h**

Micro-Instructions: **Op Code Fetch (3 Clock Cycles)**

ALU(A) <-- A; ALU(B) <-- R;
ALU(F) <-- ALU(A) - ALU(B);
Z <-- Zero Condition Bit

CMP M **Compare Indirect With A Register**

Addressing Mode: **Indirect**

Operation: **if A - MEM(R) = 0: Z <-- 1; IP <-- IP + 1**

Encoding: **00111 100 (3Ch)**

Clock Cycles: **7**

Description: **The contents of the memory address pointed to by the R Register are subtracted from the contents of the A Register. If the result is zero, the Zero Flag is set, otherwise it is reset. The result is discarded. IP increments.**

Example:

Preconditions:	**Address 10h =**	**06h**
	R Register =	**10h**
	A Register =	**06h**
	IP =	**00h**
	Z =	**don't care**
Instruction:	**CMP M**	
Postconditions:	**A Register =**	**06h**
	IP =	**01h**
	Z =	**1h**

Micro-Instructions: **Op Code Fetch (3 Clock Cycles)**

OR <-- R

AR <-- OR; Memory Address <-- AR; Assert MEMR

DR <-- Memory Data; Memory Address <-- AR; Assert MEMR

ALU(A) <-- A; ALU(B) <-- DR;
ALU(F) <-- ALU(A) - ALU(B);
Z <-- Zero Condition Bit

CMP I data **Compare Immediate With A Register**

Addressing Mode: **Immediate**

Operation: **if A - data = 0: Z <-- 1; IP <-- IP + 2**

Encoding: **Byte 1: 00111 110 (3Eh)**
Byte 2: 8-bit data

Clock Cycles: **6**

Description: **The data in byte 2 are subtracted from contents of the A Register. If the result is zero, the Zero Flag is set, otherwise it is reset. The result is discarded. IP increments 2 times.**

Example:

Preconditions:	**A Register =**	**06h**
	IP =	**00h**
	Z =	**don't care**
Instruction:	**CMP 06h**	
Postconditions:	**A Register =**	**06h**
	IP =	**02h**
	Z =	**1h**

Micro-Instructions: **Op Code Fetch (3 Clock Cycles)**

AR <-- IP; Memory Address <-- AR; Assert MEMR

DR <-- Memory Data; IP <-- IP + 1;
Memory Address <-- AR; Assert MEMR

ALU(A) <-- A; ALU(B) <-- DR;
ALU(F) <-- ALU(A) - ALU(B);
Z <-- Zero Condition Bit

DEC address **Decrement Direct & Move To A Register**

Addressing Mode: **Direct**

Operation: **A <-- MEM(address) - 1; IP <-- IP + 2**

Encoding: **Byte 1: 01110 000 (70h)**
Byte 2: 8-bit address

Clock Cycles: **11**

Description: **The contents of the memory address in byte 2 are transferred to the A Register and decremented. The result is stored in the A Register. IP increments 2 times.**

Example:

Preconditions:	**Address 10h =**	**04h**
	A Register =	**06h**
	IP =	**00h**
Instruction:	**DEC 10h**	
Postconditions:	**A Register =**	**03h**
	IP =	**02h**

Micro-Instructions: **Op Code Fetch (3 Clock Cycles)**

AR <-- IP; Memory Address <-- AR; Assert MEMR

DR <-- Memory Data; IP <-- IP + 1;
Memory Address <-- AR; Assert MEMR

OR <-- DR

AR <-- OR; Memory Address <-- AR; Assert MEMR

DR <-- Memory Data; Memory Address <-- AR; Assert MEMR

A <-- DR

ALU(A) <-- A; ALU(F) <-- ALU(A) - 1

A <-- ALU(F)

DEC A **Decrement A Register**

Addressing Mode: **Register**

Operation: **A <-- A - 1; IP <-- IP + 1**

Encoding: **01110 010 (72h)**

Clock Cycles: **4**

Description: **The contents of the A Register are decremented. The result is stored in the A Register. IP increments.**

Example:

Preconditions:	**A Register =**	**06h**
	IP =	**00h**
Instruction:	**DEC A**	
Postconditions:	**A Register =**	**05h**
	IP =	**01h**

Micro-Instructions: **Op Code Fetch (3 Clock Cycles)**

ALU(A) <-- A; ALU(F) <-- ALU(A) - 1;
A <-- ALU(F)

DEC R **Decrement R Register**

Addressing Mode: **Register**

Operation: **R <-- R - 1; IP <-- IP + 1**

Encoding: **01110 011 (73h)**

Clock Cycles: **7**

Description: **The contents of the R Register are transferred to the A Register. The A Register is decremented. The result is moved to the R Register. IP increments.**

Example:

Preconditions:	**R Register =**	**04h**
	A Register =	**06h**
	IP =	**00h**
Instruction:	**DEC R**	
Postconditions:	**R Register =**	**03h**
	A Register =	**03h**
	IP =	**01h**

Micro-Instructions: **Op Code Fetch (3 Clock Cycles)**

A <-- R

ALU(A) <-- A; ALU(F) <-- ALU(A) - 1

A <-- ALU(F)

R <-- A

DEC M — **Decrement Indirect & Move To A Register**

Addressing Mode: Indirect

Operation: A <-- MEM(R) - 1; IP <-- IP + 1

Encoding: 01110 100 (74h)

Clock Cycles: 9

Description: The contents of the memory address pointed to by the R Register are transferred to the A Register. The A Register is decremented. The result is stored in the A Register. IP increments.

Example:

Preconditions:	Address 10h =	04h
	R Register =	10h
	A Register =	06h
	IP =	00h
Instruction:	DEC M	
Postconditions:	A Register =	03h
	IP =	01h

Micro-Instructions:

Op Code Fetch (3 Clock Cycles)

OR <-- R

AR <-- OR; Memory Address <-- AR; Assert MEMR

DR <-- Memory Data; Memory Address <-- AR; Assert MEMR

A <-- DR

ALU(A) <-- A; ALU(F) <-- ALU(A) - 1

A <-- ALU(F)

DEC I data **Decrement Immediate & Move To A Register**

Addressing Mode: **Immediate**

Operation: **A <-- data - 1; IP <-- IP + 2**

Encoding: **Byte 1: 01110 110 (76h)**
Byte 2: 8-bit data

Clock Cycles: **8**

Description: **The data in byte 2 are transferred to the A Register. The A Register is decremented. The result is stored in the A Register. IP increments 2 times.**

Example:

Preconditions:	**A Register =**	**06h**
	IP =	**00h**
Instruction:	**DEC 10h**	
Postconditions:	**A Register =**	**0Fh**
	IP =	**02h**

Micro-Instructions: **Op Code Fetch (3 Clock Cycles)**

AR <-- IP; Memory Address <-- AR; Assert MEMR

DR <-- Memory Data; IP <-- IP + 1;
Memory Address <-- AR; Assert MEMR

A <-- DR

ALU(A) <-- A; ALU(F) <-- ALU(A) - 1

A <-- ALU(F)

IN address **Read Port Direct**

Addressing Mode: **Direct**

Operation: **A <-- PORT(address); IP <-- IP + 2**

Encoding: **Byte 1: 00001 000 (08h)**
Byte 2: 8-bit address

Clock Cycles: **9**

Description: **The contents of the port address in byte 2 are transferred to the A Register. IP increments 2 times.**

Example:

Preconditions:	**Address 10h =**	**04h**
	A Register =	**06h**
	IP =	**00h**
Instruction:	**IN 10h**	
Postconditions:	**A Register =**	**04h**
	IP =	**02h**

Micro-Instructions: **Op Code Fetch (3 Clock Cycles)**

AR <-- IP; Memory Address <-- AR; Assert MEMR

DR <-- Memory Data; IP <-- IP + 1;
Memory Address <-- AR; Assert MEMR

OR <-- DR

AR <-- OR; Port Address <-- AR; Assert IOR

DR <-- Port Data; Port Address <-- AR; Assert IOR

A <-- DR

IN P **Read Port Indirect**

Addressing Mode: **Indirect**

Operation: **A <-- PORT(R); IP <-- IP + 1**

Encoding: **00001 100 (0Ch)**

Clock Cycles: **7**

Description: **The contents of the port address pointed to by the R Register are transferred to the A Register. IP increments.**

Example:

Preconditions:	**Address 10h =**	**04h**
	R Register =	**10h**
	A Register =	**06h**
	IP =	**00h**
Instruction:	**IN P**	
Postconditions:	**A Register =**	**04h**
	IP =	**01h**

Micro-Instructions: **Op Code Fetch (3 Clock Cycles)**

OR <-- R

AR <-- OR; Port Address <-- AR; Assert IOR

DR <-- Port Data; Port Address <-- AR; Assert IOR

A <-- DR

INV address **Invert Direct & Move To A Register**

Addressing Mode: **Direct**

Operation: **A <-- [MEM(address)]'; IP <-- IP + 2**

Encoding: **Byte 1: 10001 000 (88h)**
Byte 2: 8-bit address

Clock Cycles: **9**

Description: **The contents of the memory address are inverted and stored in the A Register. IP increments 2 times.**

Example:

Preconditions:	**Address 10h =**	**04h**
	A Register =	**06h**
	IP =	**00h**
Instruction:	**INV 10h**	
Postconditions:	**A Register =**	**FBh**
	IP =	**02h**

Micro-Instructions: **Op Code Fetch (3 Clock Cycles)**

AR <-- IP; Memory Address <-- AR; Assert MEMR

DR <-- Memory Data; IP <-- IP + 1;
Memory Address <-- AR; Assert MEMR

OR <-- DR

AR <-- OR; Memory Address <-- AR; Assert MEMR

DR <-- Memory Data; Memory Address <-- AR;
Assert MEMR

ALU(B) <-- DR; ALU(F) <-- [ALU(B)]';
A <-- ALU(F)

INV A **Invert A Register**

Addressing Mode: **Register**

Operation: **A <-- [A]'; IP <-- IP + 1**

Encoding: **10001 010 (8Ah)**

Clock Cycles: **4**

Description: **The contents of the A Register are inverted. The result is stored in the A Register. IP increments.**

Example:

Preconditions:	**A Register =**	**04h**
	IP =	**00h**
Instruction:	**INV A**	
Postconditions:	**A Register =**	**FBh**
	IP =	**01h**

Micro-Instructions: **Op Code Fetch (3 Clock Cycles)**

ALU(B) <-- A; ALU(F) <-- [ALU(B)]';
A <-- ALU(F)

INV R **Invert R Register**

Addressing Mode: **Register**

Operation: **R <-- [R]'; IP <-- IP + 1**

Encoding: **10001 011 (8Bh)**

Clock Cycles: **5**

Description: **The contents of the R Register are inverted and the results are moved to the R Register. IP increments.**

Example:

Preconditions:	**R Register =**	**04h**
	A Register =	**06h**
	IP =	**00h**
Instruction:	**INV R**	
Postconditions:	**R Register =**	**FBh**
	A Register =	**FBh**
	IP =	**01h**

Micro-Instructions: **Op Code Fetch (3 Clock Cycles)**

ALU(B) <-- R; ALU(F) <-- [ALU(B)]';

A <-- ALU(F)

R <-- A

INV M **Invert Indirect & Move To A Register**

Addressing Mode: **Indirect**

Operation: **A <-- [MEM(R)]'; IP <-- IP + 1**

Encoding: **10001 100 (8Ch)**

Clock Cycles: **7**

Description: **The contents of the memory address pointed to by the R Register are inverted and stored in the A Register. IP increments.**

Example:

Preconditions:	**Address 10h =**	**04h**
	R Register =	**10h**
	A Register =	**06h**
	IP =	**00h**
Instruction:	**INV M**	
Postconditions:	**A Register =**	**FBh**
	IP =	**01h**

Micro-Instructions: **Op Code Fetch (3 Clock Cycles)**

OR <-- R

AR <-- OR; Memory Address <-- AR; Assert MEMR

DR <-- Memory Data; Memory Address <-- AR; Assert MEMR

ALU(B) <-- DR; ALU(F) <-- [ALU(B)]'; A <-- ALU(F)

INV I data **Invert Immediate & Move To A Register**

Addressing Mode: **Immediate**

Operation: **A <-- [data]'; IP <-- IP + 2**

Encoding: **Byte 1: 10001 110 (8Eh)**
Byte 2: 8-bit data

Clock Cycles: **6**

Description: **The data in byte 2 are inverted and stored in the A Register. IP increments 2 times.**

Example:

Preconditions:	**A Register =**	**06h**
	IP =	**00h**
Instruction:	**INV 04h**	
Postconditions:	**A Register =**	**FBh**
	IP =	**02h**

Micro-Instructions: **Op Code Fetch (3 Clock Cycles)**

AR <-- IP; Memory Address <-- AR; Assert MEMR

DR <-- Memory Data; IP <-- IP + 1;
Memory Address <-- AR; Assert MEMR

ALU(B) <-- DR; ALU(F) <-- [ALU(B)]';
A <-- ALU(F)

JMP address **Jump Direct**

Addressing Mode: **Direct**

Operation: **IP <-- address**

Encoding: **Byte 1: 00100 000 (20h)**
Byte 2: 8-bit address

Clock Cycles: **6**

Description: **The address in byte 2 is transferred to the Instruction Pointer.**

Example:

Preconditions:	**Address 10h =**	**04h**
	IP =	**00h**
Instruction:	**JMP 10h**	
Postconditions:	**IP =**	**04h**

Micro-Instructions: **Op Code Fetch (3 Clock Cycles)**

AR <-- IP; Memory Address <-- AR; Assert MEMR

DR <-- Memory Data; Memory Address <-- AR; Assert MEMR

IP <-- DR

JMP R **Jump Indirect**

Addressing Mode: **Register**

Operation: **IP <-- R**

Encoding: **00100 011 (23h)**

Clock Cycles: **4**

Description: **The contents of the R Register are transferred to the Instruction Pointer.**

Example:

Preconditions:	**R Register =**	**10h**
	IP =	**00h**
Instruction:	**JMP R**	
Postconditions:	**IP =**	**10h**

Micro-Instructions: **Op Code Fetch (3 Clock Cycles)**

IP <-- R

JNZ address **Jump Direct If Not Zero**

Addressing Mode: **Direct**

Operation: **if Z = 0: IP <-- address**

Encoding: **Byte 1: 00101 000 (28h)**
Byte 2: 8-bit address

Clock Cycles: **6 if Z = 0**
4 if Z = 1

Description: **If the Zero Flag is 0, the address in byte 2 is transferred to the Instruction Pointer. If the Zero Flag is 1, the IP is simply incremented to the next instruction.**

Example:

Preconditions:	**IP =**	**00h**
	Z =	**0h**
Instruction:	**JNZ 10h**	
Postconditions:	**IP =**	**10h**

Micro-Instructions: **Op Code Fetch (3 Clock Cycles)**

if Z = 0: JMP address

if Z = 1: IP <-- IP + 1

JNZ R **Jump If Not Zero**

Addressing Mode: **Register**

Operation: **if Z = 0: IP <-- R**

Encoding: **00101 011 (2Bh)**

Clock Cycles: **4**

Description: **If the Zero Flag is 0, the contents of the R Register are transferred to the Instruction Pointer. If the Zero Flag is 1, no operation is performed.**

Example:

Preconditions:	**IP =**	**00h**
	R =	**10h**
	Z =	**0h**
Instruction:	**JNZ R**	
Postconditions:	**IP =**	**10h**

Micro-Instructions: **Op Code Fetch (3 Clock Cycles)**

if Z = 0: JMP R

if Z = 1: No Operation

JZ address **Jump Direct If Zero**

Addressing Mode: **Direct**

Operation: **if Z = 1: IP <-- address**

Encoding: **Byte 1: 00110 000 (30h)**
Byte 2: 8-bit address

Clock Cycles: **6 if Z = 1**
4 if Z = 0

Description: **If the Zero Flag is 1, the address in byte 2 is transferred to the Instruction Pointer. If the Zero Flag is 0, the IP is simply incremented to the next instruction.**

Example:

Preconditions:	**IP =**	**00h**
	Z =	**1h**
Instruction:	**JZ 10h**	
Postconditions:	**IP =**	**10h**

Micro-Instructions: **Op Code Fetch (3 Clock Cycles)**

if Z = 1: JMP address

if Z = 0: IP <-- IP + 1

JZ R **Jump If Zero**

Addressing Mode: **Register**

Operation: **if Z = 1: IP <-- R**

Encoding: **00110 011 (33h)**

Clock Cycles: **4**

Description: **If the Zero Flag is 1, the contents of the R Register are transferred to the Instruction Pointer. If the Zero Flag is 0, no operation is performed.**

Example:

Preconditions:	**IP =**	**00h**
	R =	**10h**
	Z =	**1h**
Instruction:	**JZ R**	
Postconditions:	**IP =**	**10h**

Micro-Instructions: **Op Code Fetch (3 Clock Cycles)**

if Z = 1: JMP R

if Z = 0: No Operation

LDA address **Load A Register Direct**

Addressing Mode: **Direct**

Operation: **A <-- MEM(address); IP <-- IP + 2**

Encoding: **Byte 1: 01000 000 (40h)**
Byte 2: 8-bit address

Clock Cycles: **9**

Description: **The contents of the memory address in byte 2 are transferred to the A Register. IP increments 2 times.**

Example:

Preconditions:	**Address 10h =**	**04h**
	A Register =	**06h**
	IP =	**00h**
Instruction:	**LDA 10h**	
Postconditions:	**A Register =**	**04h**
	IP =	**02h**

Micro-Instructions: **Op Code Fetch (3 Clock Cycles)**

AR <-- IP; Memory Address <-- AR; Assert MEMR

DR <-- Memory Data; IP <-- IP + 1;
Memory Address <-- AR; Assert MEMR

OR <-- DR

AR <-- OR; Memory Address <-- AR; Assert MEMR

DR <-- Memory Data; Memory Address <-- AR;
Assert MEMR

A <-- DR

LDA A **Load A Register With A Register**

Addressing Mode: **Register**

Operation: **A <-- A; IP <-- IP + 1**

Encoding: **01000 010 (42h)**

Clock Cycles: **4**

Description: **The contents of the A Register are left in the A Register. This instruction does not change the state of the CPU, except to increment the IP.**

Example:

Preconditions:	**A Register =**	**06h**
	IP =	**00h**
Instruction:	**LDA A**	
Postconditions:	**A Register =**	**06h**
	IP =	**01h**

Micro-Instructions: **Op Code Fetch (3 Clock Cycles)**

No Operation

LDA R **Load A Register With R Register**

Addressing Mode: **Register**

Operation: **A <-- R; IP <-- IP + 1**

Encoding: **01000 011 (43h)**

Clock Cycles: **4**

Description: **The contents of the R Register are transferred to the A Register. IP increments.**

Example:

Preconditions:	**R Register =**	**04h**
	A Register =	**06h**
	IP =	**00h**
Instruction:	**LDA R**	
Postconditions:	**A Register =**	**04h**
	IP =	**01h**

Micro-Instructions: **Op Code Fetch (3 Clock Cycles)**

A <-- R

LDA M **Load A Register Indirect**

Addressing Mode: **Indirect**

Operation: **A <-- MEM(R); IP <-- IP + 1**

Encoding: **01000 100 (44h)**

Clock Cycles: **7**

Description: **The contents of the memory address pointed to by the R Register are transferred to the A Register. IP increments.**

Example:

Preconditions:	**Address 10h =**	**04h**
	R Register =	**10h**
	A Register =	**06h**
	IP =	**00h**
Instruction:	**LDA M**	
Postconditions:	**A Register =**	**04h**
	IP =	**01h**

Micro-Instructions: **Op Code Fetch (3 Clock Cycles)**

OR <-- R

AR <-- OR; Memory Address <-- AR; Assert MEMR

DR <-- Memory Data; Memory Address <-- AR; Assert MEMR

A <-- DR

LDA I data **Load A Register Immediate**

Addressing Mode: **Immediate**

Operation: **A <-- data; IP <-- IP + 2**

Encoding: **Byte 1: 01000 110 (46h)**
Byte 2: 8-bit data

Clock Cycles: **6**

Description: **The data in byte 2 are transferred to the A Register. IP increments 2 times.**

Example:

Preconditions:	**A Register =**	**06h**
	IP =	**00h**
Instruction:	**LDA 10h**	
Postconditions:	**A Register =**	**10h**
	IP =	**02h**

Micro-Instructions: **Op Code Fetch (3 Clock Cycles)**

AR <-- IP; Memory Address <-- AR; Assert MEMR

DR <-- Memory Data; IP <-- IP + 1;
Memory Address <-- AR; Assert MEMR

A <-- DR

LDR address **Load R Register Direct**

Addressing Mode: **Direct**

Operation: **R <-- MEM(address); IP <-- IP + 2**

Encoding: **Byte 1: 01001 000 (48h)**
Byte 2: 8-bit address

Clock Cycles: **9**

Description: **The contents of the memory address in byte 2 are transferred to the R Register. IP increments 2 times.**

Example:

Preconditions:	**Address 10h =**	**04h**
	R Register =	**06h**
	IP =	**00h**
Instruction:	**LDR 10h**	
Postconditions:	**R Register =**	**04h**
	IP =	**02h**

Micro-Instructions: **Op Code Fetch (3 Clock Cycles)**

AR <-- IP; Memory Address <-- AR; Assert MEMR

DR <-- Memory Data; IP <-- IP + 1;
Memory Address <-- AR; Assert MEMR

OR <-- DR

AR <-- OR; Memory Address <-- AR; Assert MEMR

DR <-- Memory Data; Memory Address <-- AR;
Assert MEMR

R <-- DR

LDR A **Load R Register With A Register**

Addressing Mode: **Register**

Operation: **R <-- A; IP <-- IP + 1**

Encoding: **01001 010 (4Ah)**

Clock Cycles: **4**

Description: **The contents of the A Register are transferred to the R Register. IP increments.**

Example:

Preconditions:	**R Register =**	**04h**
	A Register =	**06h**
	IP =	**00h**
Instruction:	**LDR A**	
Postconditions:	**A Register =**	**04h**
	IP =	**01h**

Micro-Instructions: **Op Code Fetch (3 Clock Cycles)**

R <-- A

LDR R **Load R Register With R Register**

Addressing Mode: **Register**

Operation: **R <-- R; IP <-- IP + 1**

Encoding: **01001 011 (4Bh)**

Clock Cycles: **4**

Description: **The contents of the R Register are left in the R Register. This instruction does not change the state of the CPU, except to increment the IP.**

Example:

Preconditions:	**R Register =**	**06h**
	IP =	**00h**
Instruction:	**LDR R**	
Postconditions:	**R Register =**	**06h**
	IP =	**01h**

Micro-Instructions: **Op Code Fetch (3 Clock Cycles)**

No Operation

LDR M **Load R Register Indirect**

Addressing Mode: **Indirect**

Operation: **R <-- MEM(R); IP <-- IP + 1**

Encoding: **01001 100 (4Ch)**

Clock Cycles: **7**

Description: **The contents of the memory address pointed to by the R Register are transferred to the R Register. IP increments.**

Example:

Preconditions:	**Address 10h =**	**04h**
	R Register =	**10h**
	IP =	**00h**
Instruction:	**LDR M**	
Postconditions:	**R Register =**	**04h**
	IP =	**01h**

Micro-Instructions: **Op Code Fetch (3 Clock Cycles)**

OR <-- R

AR <-- OR; Memory Address <-- AR; Assert MEMR

DR <-- Memory Data; Memory Address <-- AR; Assert MEMR

R <-- DR

LDR I data **Load R Register Immediate**

Addressing Mode: **Immediate**

Operation: **R <-- data; IP <-- IP + 2**

Encoding: **Byte 1: 01001 110 (4Eh)**
Byte 2: 8-bit data

Clock Cycles: **6**

Description: **The data in byte 2 are transferred to the R Register. IP increments 2 times.**

Example:

Preconditions:	**R Register =**	**06h**
	IP =	**00h**
Instruction:	**LDR 10h**	
Postconditions:	**R Register =**	**10h**
	IP =	**02h**

Micro-Instructions: **Op Code Fetch (3 Clock Cycles)**

AR <-- IP; Memory Address <-- AR; Assert MEMR

DR <-- Memory Data; IP <-- IP + 1;
Memory Address <-- AR; Assert MEMR

R <-- DR

OR address **OR Direct With A Register**

Addressing Mode: **Direct**

Operation: **A <-- A ^ MEM(address); IP <-- IP + 2**

Encoding: **Byte 1: 10000 000 (80h)**
Byte 2: 8-bit address

Clock Cycles: **9**

Description: **The contents of the memory address in byte 2 are ORed with the contents of the A Register. The result is stored in the A Register. IP increments 2 times.**

Example:

Preconditions:	**Address 10h =**	**40h**
	A Register =	**06h**
	IP =	**00h**
Instruction:	**OR 10h**	
Postconditions:	**A Register =**	**46h**
	IP =	**02h**

Micro-Instructions: **Op Code Fetch (3 Clock Cycles)**

AR <-- IP; Memory Address <-- AR; Assert MEMR

DR <-- Memory Data; IP <-- IP + 1;
Memory Address <-- AR; Assert MEMR

OR <-- DR

AR <-- OR; Memory Address <-- AR; Assert MEMR

DR <-- Memory Data; Memory Address <-- AR;
Assert MEMR

ALU(A) <-- A; ALU(B) <-- DR;
ALU(F) <-- ALU(A) ^ ALU(B);
A <-- ALU(F)

OR A — **OR A Register With A Register**

Addressing Mode: Register

Operation: A <-- A ^ A; IP <-- IP + 1

Encoding: 10000 010 (82h)

Clock Cycles: 4

Description: The contents of the A Register are ORed with the contents of the A Register. The result is stored in the A Register. IP increments.

Example:

Preconditions:	**A Register =**	**06h**
	IP =	**00h**
Instruction:	**OR A**	
Postconditions:	**A Register =**	**06h**
	IP =	**01h**

Micro-Instructions: Op Code Fetch (3 Clock Cycles)

ALU(A) <-- A; ALU(B) <-- A;
ALU(F) <-- ALU(A) ^ ALU(B);
A <-- ALU(F)

OR R — **OR R Register With A Register**

Addressing Mode: Register

Operation: A <-- A ^ R; IP <-- IP + 1

Encoding: 10000 011 (83h)

Clock Cycles: 4

Description: The contents of the R Register are ORed with the contents of the A Register. The result is stored in the A Register. IP increments.

Example:

Preconditions:	**R Register =**	**40h**
	A Register =	**06h**
	IP =	**00h**
Instruction:	**OR R**	
Postconditions:	**A Register =**	**46h**
	IP =	**01h**

Micro-Instructions: Op Code Fetch (3 Clock Cycles)

```
ALU(A) <-- A; ALU(B) <-- R;
ALU(F) <-- ALU(A) ^ ALU(B);
A <-- ALU(F)
```

OR M **OR Indirect With A Register**

Addressing Mode: **Indirect**

Operation: **A <-- A ^ MEM(R); IP <-- IP + 1**

Encoding: **10000 100 (84h)**

Clock Cycles: **7**

Description: **The contents of the memory address pointed to by the R Register are ORed with the contents of the A Register. The result is stored in the A Register. IP increments.**

Example:

Preconditions:	**Address 10h =**	**40h**
	R Register =	**10h**
	A Register =	**06h**
	IP =	**00h**
Instruction:	**OR M**	
Postconditions:	**A Register =**	**46h**
	IP =	**01h**

Micro-Instructions: **Op Code Fetch (3 Clock Cycles)**

OR <-- R

AR <-- OR; Memory Address <-- AR; Assert MEMR

DR <-- Memory Data; Memory Address <-- AR; Assert MEMR

ALU(A) <-- A; ALU(B) <-- DR;
ALU(F) <-- ALU(A) ^ ALU(B);
A <-- ALU(F)

OR I data — **OR Immediate With A Register**

Addressing Mode: Immediate

Operation: A <-- A ^ data; IP <-- IP + 2

Encoding: Byte 1: 10000 110 (86h)
Byte 2: 8-bit data

Clock Cycles: 6

Description: The data in byte 2 are ORed with the contents of the A Register. The result is stored in the A Register. IP increments 2 times.

Example:

Preconditions:	A Register =	06h
	IP =	00h
Instruction:	OR 10h	
Postconditions:	A Register =	16h
	IP =	02h

Micro-Instructions: Op Code Fetch (3 Clock Cycles)

AR <-- IP; Memory Address <-- AR; Assert MEMR

DR <-- Memory Data; IP <-- IP + 1;
Memory Address <-- AR; Assert MEMR

ALU(A) <-- A; ALU(B) <-- DR;
ALU(F) <-- ALU(A) ^ ALU(B);
A <-- ALU(F)

OUT address — **Write Port Direct**

Addressing Mode: Direct

Operation: PORT(address) <-- A; IP <-- IP + 2

Encoding: Byte 1: 00010 000 (10h)
Byte 2: 8-bit address

Clock Cycles: 9

Description: The contents of the A Register are transferred to the Port address in byte 2. IP increments 2 times.

Example:

Preconditions:	Address 10h =	04h
	A Register =	06h
	IP =	00h
Instruction:	OUT 10h	
Postconditions:	Address 10h =	06h
	IP =	02h

Micro-Instructions: Op Code Fetch (3 Clock Cycles)

AR <-- IP; Memory Address <-- AR; Assert MEMR

DR <-- Memory Data; IP <-- IP + 1;
Memory Address <-- AR; Assert MEMR

OR <-- DR

AR <-- OR; Port Address <-- AR; DR <-- A;
Port Data <-- DR

Port Address <-- AR; Port <-- DR; Assert IOW

Port Address <-- AR; Port Data <-- DR

OUT P **Write Port Indirect**

Addressing Mode: **Indirect**

Operation: **PORT(R) <-- A; IP <-- IP + 1**

Encoding: **00010 100 (14h)**

Clock Cycles: **7**

Description: **The contents of the A Register are transferred to the port address pointed to by the R Register. IP increments.**

Example:

Preconditions:	**Address 10h =**	**04h**
	R Register =	**10h**
	A Register =	**06h**
	IP =	**00h**
Instruction:	**OUT P**	
Postconditions:	**Address 10h =**	**06h**
	IP =	**01h**

Micro-Instructions: **Op Code Fetch (3 Clock Cycles)**

OR <-- R

AR <-- OR; Port Address <-- AR; DR <-- A;
Port Data <-- DR

Port Address <-- AR; Port Data <-- DR; Assert IOW

Port Address <-- AR; Port Data <-- DR

SHL address **Shift Left Direct & Move To A Register**

Addressing Mode: **Direct**

Operation: **A <-- $[MEM(address)]_{6..0}$ ## 0; IP <-- IP + 2**

Encoding: **Byte 1: 10010 000 (90h)**
Byte 2: 8-bit address

Clock Cycles: **11**

Description: **The contents of the memory address are transferred to the A Register and shifted 1 bit left. Zero is shifted into the LSB. The result is stored in the A Register. IP increments twice.**

Example:

Preconditions:	**Address 10h =**	**04h**
	A Register =	**06h**
	IP =	**00h**
Instruction:	**SHL 10h**	
Postconditions:	**A Register =**	**08h**
	IP =	**02h**

Micro-Instructions: **Op Code Fetch (3 Clock Cycles)**

AR <-- IP; Memory Address <-- AR; Assert MEMR

DR <-- Memory Data; IP <-- IP + 1;
Memory Address <-- AR; Assert MEMR

OR <-- DR

AR <-- OR; Memory Address <-- AR; Assert MEMR

DR <-- Memory Data; Memory Address <-- AR; Assert MEMR

A <-- DR

ALU(A) <-- A; ALU(F) <-- $[ALU(A)]_{6..0}$ ## 0

A <-- ALU(F)

SHL A **Shift Left A Register**

Addressing Mode: **Register**

Operation: **A <-- $[A]_{6..0}$ ## 0; IP <-- IP + 1**

Encoding: **10010 010 (92h)**

Clock Cycles: **4**

Description: **The contents of the A Register are shifted 1 bit to the left. A zero is shifted into the LSB. The result is stored in the A Register. IP increments.**

Example:

Preconditions:	**A Register =**	**04h**
	IP =	**00h**
Instruction:	**SHL A**	
Postconditions:	**A Register =**	**08h**
	IP =	**01h**

Micro-Instructions: **Op Code Fetch (3 Clock Cycles)**

ALU(A) <-- A; ALU(F) <-- $[ALU(A)]_{6..0}$ ## 0;
A <-- ALU(F)

SHL R **Shift Left R Register**

Addressing Mode: **Register**

Operation: **R <-- $[R]_{6..0}$ ## 0; IP <-- IP + 1**

Encoding: **10010 011 (93h)**

Clock Cycles: **7**

Description: **The contents of the R Register transferred to the A Register and shifted 1 bit to the left. A zero is shifted into the LSB. The result is moved to the R Register. IP increments.**

Example:

Preconditions:	**R Register =**	**04h**
	A Register =	**06h**
	IP =	**00h**
Instruction:	**SHL R**	
Postconditions:	**R Register =**	**08h**
	A Register =	**08h**
	IP =	**01h**

Micro-Instructions: **Op Code Fetch (3 Clock Cycles)**

A <-- R

ALU(A) <-- R; ALU(F) <-- $[ALU(A)]_{6..0}$ ## 0

A <-- ALU(F)

R <-- A

SHL M **Shift Left Indirect & Move To A Register**

Addressing Mode: **Indirect**

Operation: **A <-- $[MEM(R)]_{6..0}$ ## 0; IP <-- IP + 1**

Encoding: **10010 100 (94h)**

Clock Cycles: **9**

Description: **The contents of the memory address pointed to by the R Register are transferred into the A Register and shifted 1 bit to the left. A zero is shifted into the LSB. The result is stored in the A Register. IP increments.**

Example:

Preconditions:	**Address 10h =**	**04h**
	R Register =	**10h**
	A Register =	**06h**
	IP =	**00h**
Instruction:	**SHL M**	
Postconditions:	**A Register =**	**08h**
	IP =	**01h**

Micro-Instructions: **Op Code Fetch (3 Clock Cycles)**

OR <-- R

AR <-- OR; Memory Address <-- AR; Assert MEMR

DR <-- Memory Data; Memory Address <-- AR; Assert MEMR

A <-- DR

ALU(A) <-- A; ALU(F) <-- $[ALU(A)]_{6..0}$ ## 0

A <-- ALU(F)

SHL I data **Shift Left Immediate & Move To A Register**

Addressing Mode: **Immediate**

Operation: **A <-- $[data]_{6..0}$ ## 0; IP <-- IP + 2**

Encoding: **Byte 1: 10010 110 (96h)**
Byte 2: 8-bit data

Clock Cycles: **8**

Description: **The data in byte 2 are transferred to the A Register and shifted 1 bit to the left. A zero is shifted into the LSB. The result is stored in the A Register. IP increments 2 times.**

Example:

Preconditions:	**A Register =**	**06h**
	IP =	**00h**
Instruction:	**SHL 04h**	
Postconditions:	**A Register =**	**08h**
	IP =	**02h**

Micro-Instructions: **Op Code Fetch (3 Clock Cycles)**

AR <-- IP; Memory Address <-- AR; Assert MEMR

DR <-- Memory Data; IP <-- IP + 1;
Memory Address <-- AR; Assert MEMR

A <-- DR

ALU(A) <-- A; ALU(F) <-- $[ALU(A)]_{6..0}$ ## 0

A <-- ALU(F)

STA address **Store A Register Direct**

Addressing Mode: **Direct**

Operation: **MEM(address) <-- A; IP <-- IP + 2**

Encoding: **Byte 1: 01010 000 (50h)**
Byte 2: 8-bit address

Clock Cycles: **9**

Description: **The contents of the A Register are transferred to the Memory address in byte 2. IP increments 2 times.**

Example:

Preconditions:	**Address 10h =**	**04h**
	A Register =	**06h**
	IP =	**00h**
Instruction:	**STA 10h**	
Postconditions:	**Address 10h =**	**06h**
	IP =	**02h**

Micro-Instructions: **Op Code Fetch (3 Clock Cycles)**

AR <-- IP; Memory Address <-- AR; Assert MEMR

DR <-- Memory Data; IP <-- IP + 1;
Memory Address <-- AR; Assert MEMR

OR <-- DR

AR <-- OR; Memory Address <-- AR;
DR <-- A; Memory Data <-- DR

Memory Address <-- AR; Memory Data <-- DR;
Assert MEMW

Memory Address <-- AR; Memory Data <-- DR

STA M **Store A Register Indirect**

Addressing Mode: **Indirect**

Operation: **MEM(R) <-- A; IP <-- IP + 1**

Encoding: **01010 100 (54h)**

Clock Cycles: **7**

Description: **The contents of the A Register are transferred to the Memory address pointed to by the R Register. IP increments.**

Example:

Preconditions:

Address 10h =	**04h**
R Register =	**10h**
A Register =	**06h**
IP =	**00h**

Instruction: **STA M**

Postconditions:

Address 10h =	**06h**
IP =	**01h**

Micro-Instructions: **Op Code Fetch (3 Clock Cycles)**

OR <-- R

AR <-- OR; Memory Address <-- AR;
DR <-- A; Memory Data <-- DR

Memory Address <-- AR; Memory Data <-- DR;
Assert MEMW

Memory Address <-- AR; Memory Data <-- DR

STR address **Store R Register Direct**

Addressing Mode: **Direct**

Operation: **MEM(address) <-- R; IP <-- IP + 2**

Encoding: **Byte 1: 01011 000 (58h)**
Byte 2: 8-bit address

Clock Cycles: **9**

Description: **The contents of the R Register are transferred to the Memory address in byte 2. IP increments 2 times.**

Example:

Preconditions:	**Address 10h =**	**04h**
	R Register =	**06h**
	IP =	**00h**
Instruction:	**STR 10h**	
Postconditions:	**Address 10h =**	**06h**
	IP =	**02h**

Micro-Instructions: **Op Code Fetch (3 Clock Cycles)**

AR <-- IP; Memory Address <-- AR; Assert MEMR

DR <-- Memory Data; IP <-- IP + 1;
Memory Address <-- AR; Assert MEMR

OR <-- DR

AR <-- OR; Memory Address <-- AR;
DR <-- R; Memory Data <-- DR

Memory Address <-- AR; Memory Data <-- DR;
Assert MEMW

Memory Address <-- AR; Memory Data <-- DR

STR M **Store R Register Indirect**

Addressing Mode: **Indirect**

Operation: **MEM(R) <-- R; IP <-- IP + 1**

Encoding: **01011 100 (5Ch)**

Clock Cycles: **7**

Description: **The contents of the R Register are transferred to the Memory address pointed to by the R Register. IP increments.**

Example:

Preconditions:
Address 10h = 04h
R Register = 10h
A Register = 06h
IP = 00h

Instruction: **STR M**

Postconditions:
Address 10h = 10h
IP = 01h

Micro-Instructions: **Op Code Fetch (3 Clock Cycles)**

OR <-- R

AR <-- OR; Memory Address <-- AR;
DR <-- R; Memory Data <-- DR

Memory Address <-- AR; Memory Data <-- DR;
Assert MEMW

Memory Address <-- AR; Memory Data <-- DR

SUB address **Subtract Direct From A Register**

Addressing Mode: **Direct**

Operation: **A <-- A - MEM(address); IP <-- IP + 2**

Encoding: **Byte 1: 01101 000 (68h)**
Byte 2: 8-bit address

Clock Cycles: **9**

Description: **The contents of the memory address in byte 2 are subtracted from the contents of the A Register. The result is stored in the A Register. IP increments 2 times.**

Example:

Preconditions:	**Address 10h =**	**04h**
	A Register =	**06h**
	IP =	**00h**
Instruction:	**SUB 10h**	
Postconditions:	**A Register =**	**02h**
	IP =	**02h**

Micro-Instructions: **Op Code Fetch (3 Clock Cycles)**

AR <-- IP; Memory Address <-- AR; Assert MEMR

DR <-- Memory Data; IP <-- IP + 1;
Memory Address <-- AR; Assert MEMR

OR <-- DR

AR <-- OR; Memory Address <-- AR; Assert MEMR

DR <-- Memory Data; Memory Address <-- AR;
Assert MEMR

ALU(A) <-- A; ALU(B) <-- DR;
ALU(F) <-- ALU(A) - ALU(B);
A <-- ALU(F)

SUB A **Subtract A Register From A Register**

Addressing Mode: **Register**

Operation: **A <-- A - A; IP <-- IP + 1**

Encoding: **01101 010 (6Ah)**

Clock Cycles: **4**

Description: **The contents of the A Register are subtracted from the contents of the A Register. The result (always zero) is stored in the A Register. IP increments.**

Example:

Preconditions:	**A Register =**	**06h**
	IP =	**00h**
Instruction:	**SUB A**	
Postconditions:	**A Register =**	**00h**
	IP =	**01h**

Micro-Instructions: **Op Code Fetch (3 Clock Cycles)**

ALU(A) <-- A; ALU(B) <-- A;
ALU(F) <-- ALU(A) - ALU(B);
A <-- ALU(F)

SUB R **Subtract R Register From A Register**

Addressing Mode: **Register**

Operation: **A <-- A - R; IP <-- IP + 1**

Encoding: **01101 011 (6Bh)**

Clock Cycles: **4**

Description: **The contents of the R Register are subtracted from the contents of the A Register. The result is stored in the A Register. IP increments.**

Example:

Preconditions:	**R Register =**	**04h**
	A Register =	**06h**
	IP =	**00h**
Instruction:	**SUB R**	
Postconditions:	**A Register =**	**02h**
	IP =	**01h**

Micro-Instructions: **Op Code Fetch (3 Clock Cycles)**

ALU(A) <-- A; ALU(B) <-- R;
ALU(F) <-- ALU(A) - ALU(B);
A <-- ALU(F)

SUB M **Subtract Indirect From A Register**

Addressing Mode: **Indirect**

Operation: **A <-- A - MEM(R); IP <-- IP + 1**

Encoding: **01101 100 (6Ch)**

Clock Cycles: **7**

Description: **The contents of the memory address pointed to by the R Register are subtracted from the contents of the A Register. The result is stored in the A Register. IP increments.**

Example:

Preconditions:	**Address 10h =**	**04h**
	R Register =	**10h**
	A Register =	**06h**
	IP =	**00h**
Instruction:	**SUB M**	
Postconditions:	**A Register =**	**02h**
	IP =	**01h**

Micro-Instructions: **Op Code Fetch (3 Clock Cycles)**

OR <-- R

AR <-- OR; Memory Address <-- AR; Assert MEMR

DR <-- Memory Data; Memory Address <-- AR; Assert MEMR

ALU(A) <-- A; ALU(B) <-- DR;
ALU(F) <-- ALU(A) - ALU(B);
A <-- ALU(F)

SUB I data — **Subtract Immediate From A Register**

Addressing Mode: Immediate

Operation: A <-- A - data; IP <-- IP + 2

Encoding: Byte 1: 01101 110 (6Eh)
Byte 2: 8-bit data

Clock Cycles: 6

Description: The data in byte 2 are subtracted from the contents of the A Register. The result is stored in the A Register. IP increments 2 times.

Example:

Preconditions:	A Register =	26h
	IP =	00h
Instruction:	SUB 10h	
Postconditions:	A Register =	16h
	IP =	02h

Micro-Instructions: Op Code Fetch (3 Clock Cycles)

AR <-- IP; Memory Address <-- AR; Assert MEMR

DR <-- Memory Data; IP <-- IP + 1;
Memory Address <-- AR; Assert MEMR

ALU(A) <-- A; ALU(B) <-- DR;
ALU(F) <-- ALU(A) - ALU(B);
A <-- ALU(F)