Homework Ch.4 Part 2

John Gangemi
U6871-4612
CDA 3103

$\boxed{1}$ End-of-Chapter Exercises

1) What are the main functions of the CPU?

⇒ The CPU is responsible for fetching program instructions, decoding each instruction that is fetched, and performing the indicated sequence of operations on the correct data (ie. the Fetch-Decode-Execute cycle).
The CPU consists of two main components, the datapath & the control unit. Together, these components carry out the Fetch-Decode-Execute cycle.

- Datapath : network of registers and an ALU connected by buses or common pathways where the timing is controlled by clocks.
- Control unit : responsible for sequencing operations and organizing data in the correct places at the correct time.

2) How is the ALU related to the CPU? What are its main functions?

⇒ The ALU belongs to the Datapath portion of the CPU. Its main functions are to carry out logic operations & arithmetic operations required during program execution.

9) Suppose that a 2M × 16 main memory is built using 256K × 8 RAM chips and memory is word addressable. (Assuming 1 word = 16 bits)

A) How many RAM chips are necessary?

⇒ Total memory capacity: [2M × 16]

- length = $2M = 2 \cdot 2^{20} = 2^{21}$ items
- width = 16 bits = 1 word per item
- Total words = length × width = $2^{21}\ \cancel{\text{items}} \times \frac{1\text{ word}}{\cancel{\text{item}}} = 2^{21}$ words of memory

Total memory capacity: [256K × 8] RAM

- length = $256K = 2^8 \cdot 2^{10} = 2^{18}$ items per chip
- width = 8 bits = $\frac{1}{2}$ word per item
- Words per chip = length × width = $\frac{2^{18}\ \cancel{\text{items}}}{\text{chip}} \times \frac{1\text{ word}}{2\text{ item}} = 2^{18}/2^1 = 2^{17}$ words per chip

- # of chips = $\frac{\text{Total words}}{\text{words per chip}} = \frac{2^{21}\ \cancel{\text{words}}}{2^{17}\ \cancel{\text{words}}\text{ per chip}} = 2^4 = \boxed{16\text{ chips}}$

B) How many RAM chips are there per memory word?

⇒ Since there is 16 bits per memory word and each RAM chip's width is 8 bits then $2^4 / 2^3 = 2^1$ chips = 2 RAM chips

C) How many address bits are needed for each RAM chip?

⇒ The length of a RAM chip = 256K = 2^{18} items = 2^N

where N = address bits = 18 bits for address

D) How many banks will this memory have?

⇒ Each 256K × 8 RAM chip is part of a 256K × 16 memory bank, therefore the number of banks is given by...

Length of memory / Length of RAM = $2M / 256K = 2^{21}/2^{18} = 2^3 = $ 8 banks

E) How many address bits are needed for all of memory?

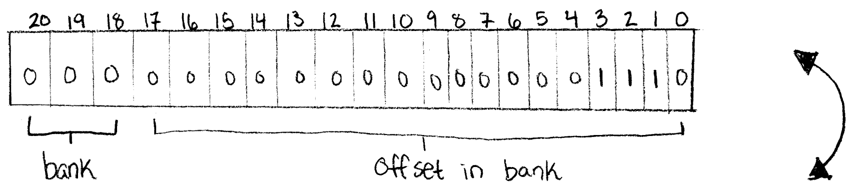
⇒ From part A, Length of main memory = 2M = 2^{21} addressable items = 2^N

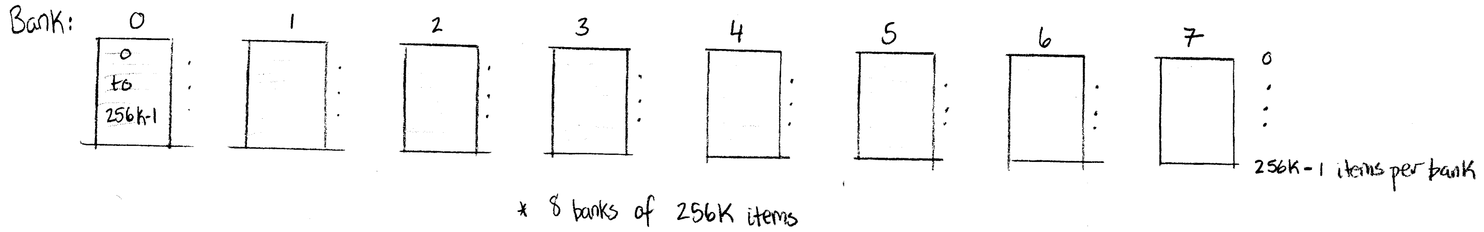
where N = address bits

= 21 bits for address

F) If high-order interleaving is used, where would address 14 (E in hex) be located?

⇒ 21 bits for address, 18 bits for offset, & 3 bits to determine chip





* 8 banks of 256K items

• Either representation of highorder interleaving shows that address 14's location would be found in Bank 0 and at the 14th offset

G) If low-order interleaving is used, where would address 14 (E in Hex) be located?

⇒ 21 bits for address, 18 bits for offset, & 3 bits to determine bank

20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0

offset in bank

bank

• Bank = $110_2 = 6$ & offset = 1

12) Computer has memory unit with 32 bits per word. Instruction set consists of 110 different operations. All instructions have an opcode and two address fields; one for memory address & one for register address. System includes 8 general-purpose registers that may be loaded directly from memory and memory may be updated directly from the registers. Direct memory to memory data movement operations are not supported. Each instruction is stored in one word of memory.

A) How many bits needed for the opcode?

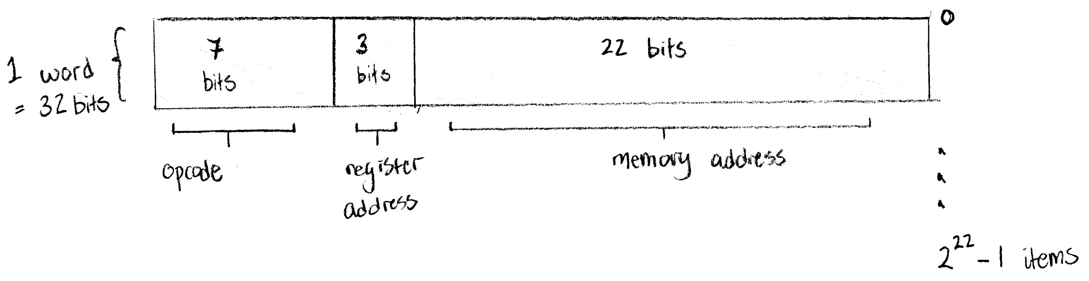
⇒ Given 110 operations in the instruction set, one would need 2^7 capacity or 7 bits for opcode

B) How many bits needed to specify the register?

⇒ Want the bits for register address...
Given the system has 8 registers, one would need 2^3 addresses or 3 bits for register address

C) How many bits are left for the memory address part of the instruction?

⇒ Since a single instruction = 1word = 32 bits, then
$31 - 7 - 3 =$ 22 bits for memory address



$2^{22} - 1$ items

D) What is the maximum allowable size for memory?

⇒ Want length x width, assuming that memory is word addressable ...

• length = 2^N items where N = bits for memory address = 22 bits

= 2^{22} = $2^2 \cdot 2^{20}$ = 4M

• width = # of words per item = 1 word per item = 32 bits

• Max size for memory is 4M x 32

E) What is the largest unsigned binary number that can be accommodated in word of memory?

⇒ 1 word = 32 bits thus $2^{32}-1$ = $\boxed{4,294,967,295}$

$(1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111)_2 = 2^N - 1 = 2^{32}-1$

13) Assume a 2^{20} byte memory

A) What are the lowest and highest addresses if memory is byte addressable?

⇒ Given the length = 2^{20} then $\boxed{0 \text{ to } 2^{20}-1}$ is the number of addressable items. With lowest = $(00000000000000000000)_2$ and highest = $(11111111111111111111)_2$ * 20 bit addresses

B) What are the lowest and highest addresses if memory is word addressable, assuming a 16-bit word?

⇒ 1 word = 16 bits = 2 bytes then length = 2^{20} ~~bytes~~ x $\frac{1 \text{ word}}{2 \text{ bytes}}$ = $2^{20}/2^1 = 2^{19}$ items

• Given length = 2^{19} then $\boxed{0 \text{ to } 2^{19}-1}$ is the number of addressable items. With lowest = $(0000000000000000000)_2$ and highest = $(1111111111111111111)_2$

* 19 bit addresses

C) What are the lowest and highest addresses if memory is word addressable, assuming a 32-bit word?

⇒ 1 word = 32 bits = 4 bytes then length = 2^{20} bytes x $\frac{1 \text{ word}}{4 \text{ bytes}}$ = $2^{20}/2^2 = 2^{18}$ items

• Given length = 2^{18} then $\boxed{0 \text{ to } 2^{18}-1}$ is the number of addressable items. With lowest = $(000000000000000000)_2$ and highest = $(111111111111111111)_2$

* 18 bit addresses

16) Explain the steps in the fetch-decode-execute cycle. Include what is happening in the various registers...

⇒ Fetch

- Copy contents of the PC to the MAR; that is the address of the next instruction to be executed is copied into the memory address register.

  MAR ← PC

- Fetch the instruction found in memory at the address given by the memory address register and place the instruction that includes an opcode and an address to the opcode's argument into the instruction register.

  IR ← M[MAR]

- Increment the program counter so that it points to the address of the next instruction.

  PC ← PC + 1

Decode

- Copy the first 12 bits of the instruction register value, the instruction's address, into the memory address register

  MAR ← IR[11-0]

- Decode the last 4 bits of the instruction register, the instruction's opcode, to get the machine instruction to perform.

  Decode IR[15-12]

Execute

- If the instruction involves data that needs to be operated on then fetch the data from its place in memory using the address in the memory address register and place the data into the memory buffer register or possibly the accumulator.

  MBR ← M[MAR]

- Execute the actual Instruction

18) Explain why, in MARIE, the MAR is only 12 bits wide and the AC is 16 bits wide.

⇒ The MAR (memory address register) is used to store the memory address of data being referenced by the current instruction. In MARIE the instruction size is 1 word or 16 bits with 12 bits for the address of an instruction, and since only 12 bits need to be stored when referencing data for an instruction, the MAR is made to accomodate 12 bits.

The AC (accumulator register) is a general-purpose register that holds data the CPU needs to process. In MARIE the instruction size is 1 word or 16 bits, and since the AC needs to handle the maximum size of an instruction it was made to accomodate 16 bits.

21) Consider the MARIE program...

Hex Address	Label	Instruction
100	Start,	LOAD A
101		ADD B
102		STORE D
103		CLEAR
104		OUTPUT
105		ADDI D
106		STORE B
107		HALT
108	A,	HEX 00FC
109	B,	DEC 14
10A	C,	HEX 0108
10B	D,	HEX 0000

Hex to Binary

0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

A) List the hexidecimal code for each instruction

LOAD A :	$(0001)_2$	⇒	1 [A]	=	1108
ADD B :	$(0011)_2$	⇒	3 [B]	=	3109
STORE D :	$(0010)_2$	⇒	2 [D]	=	210B
CLEAR :	$(1010)_2$	⇒	A	=	A000
OUTPUT :	$(0110)_2$	⇒	6	=	6000
ADDI D :	$(1011)_2$	⇒	B [D]	=	B10B
STORE B :	$(0010)_2$	⇒	2 [B]	=	2109
HALT :	$(0111)_2$	⇒	7	=	7000

Opcode

hexidecimal code

B) Draw the symbol table

Symbol	Address
Start	100
A	108
B	109
C	10A
D	10B

c) What is the value in the AC when the program terminates ?

⇒ Given that at hex address 105, for the ADDI D instruction, the value of $A = 00FC_{16}$, $B = 14_{10}$, $C = 0108_{16}$, & $D = 010A_{16}$

• Thus, after ADDI D the AC contains the decimal value 264

23) Given the instruction set for MARIE, do the following.
Decipher the following MARIE machine language instructions (write the assembly language equivalent)

* Opcode = bits 12-15 , operand = bits 0 to 11 = 3 bit hex address

A) 0010 0000 0000 0111 ⇒ opcode: 0010_2 = Store & operand : $7_{10} = 007_{16}$

• Store 007_{16}

B) 1001 0000 0000 1011 ⇒ opcode : 1001_2 = Jump & operand : $11_{10} = 00B_{16}$

• Jump $00B_{16}$

C) 0011 0000 0000 1001 ⇒ opcode : 0011_2 = Add & operand : $9_{10} = 009_{16}$

• Add 009_{16}

26) Write the following code segment in MARIE's assembley language

```
if x > 1 then
    Y = X + X;
    X = 0;
end if ;
Y = Y + 1 ;
```

⇒

Address	Label	Instruction
100		LOAD X
101		SUBT ONE
102	If,	SKIPCOND 800
103		JUMP ELSE
104		ADD ONE
105		ADD X
106		STORE Y
107		CLEAR
108		STORE X
109	Else,	LOAD Y
10A		ADD ONE
10B		STORE Y
10C		HALT
10D	ONE,	DEC 1
10E	X,	DEC 5
10F	Y,	DEC 10

Gave arbitrary values for X & Y

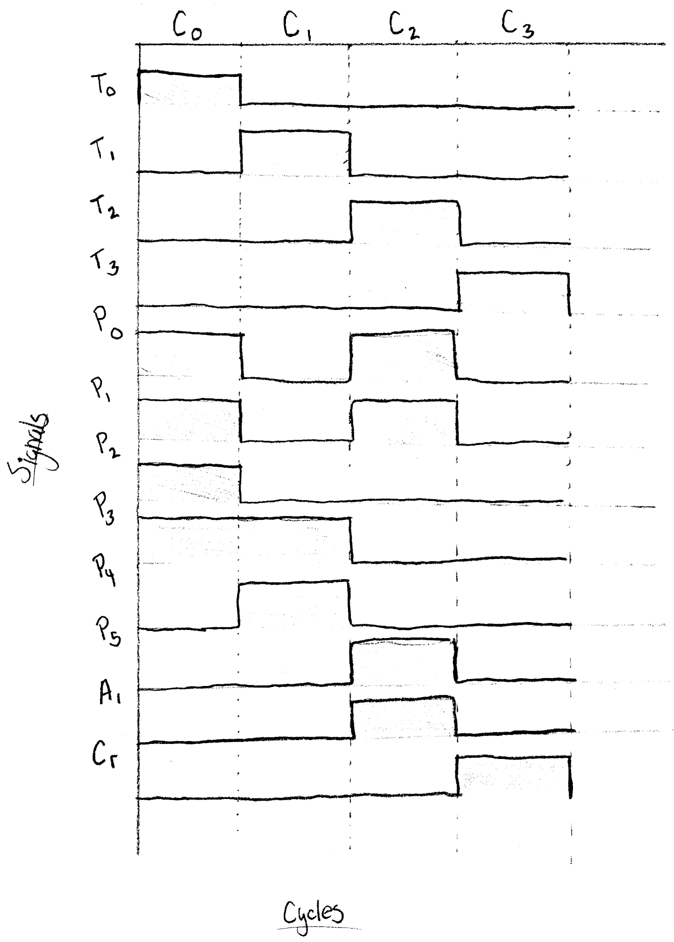
35) MARIE saves the return address for a subroutine in memory, at a location designated by the JnS instruction. In some architectures, this address is stored in a register, and in many it is stored on a stack. Which of these methods would best handle recursion?

⇒ Storing the return address on the stack (or "pushing") using the JnS instruction would allow for the "completed" recursive subroutine to remove the last (or "pop") value for the address to return, from the top of the stack. A psuedo-rewind using addresses allows the program to resume normally from a recursive function.

48) Draw the timing diagram for MARIE's Subt instruction using the format of Figure 4.16

RTL for Subt X [0100]

MAR $\leftarrow$ IR[11-0]	Read (111) Write (001)	T_0 $P_0 P_1 P_2$ P_3
MBR $\leftarrow$ M[MAR]	Read (000) Write (011)	T_1 $P_3 P_4$
AC $\leftarrow$ AC - MBR	Read (011) Write (100)	T_2 $P_0 P_1$ P_5 A_1
reset cycle counter		T_3 C_r



* shaded regions correspond to the signal being "high" otherwise it is "low"

50) Using the coding in table 4.9, translate into binary the mnemonic microcode instructions given in figure 4.23 for the 1st nine lines of the table (fetch-decode-execute cycle)

Address (7 bits)	MicroOp 1 (5 bits)	MicroOp 2 (5 bits)	Jump	Dest (7 bits)
0000000	01010	00000	0	0000000
0000001	00110	00000	0	0000000
0000010	10001	00000	0	0000000
0000011	01000	00000	0	0000000
0000100	11000	00000	1	0100000
0000101	11000	00010	1	0100111
0000110	11000	00100	1	0101010
0000111	11000	00110	1	0101100
0001000	11000	01000	1	0101111

53) Using Figure 4.23, write the binary microcode for MARIE's Add instruction. Assume the microcode begins at instruction line number 0110100_2.

Address	MicroOp1	MicroOp2	Jump	Dest
0000000	01010	00000	0	0000000
0000001	00110	00000	0	0000000
0000010	10001	00000	0	0000000
0000011	01000	00000	0	0000000
... 0000111	11000	00110	1	0110100
... 0110100	01011	00000	0	0000000
0110101	01101	00000	0	0000000
0110110	00100	00000	1	0000000

$\boxed{2}$ True / False

1) If a computer uses hardwired control, the microprogram determines the instruction set for the machine. This instruction set can never be changed unless the architecture is redesigned.

⇒ False, microprograms are not used for hardwired control

2) A branch instruction changes the flow of information by changing the PC.

⇒ True, whether unconditional or conditional branching the PC is manipulated

3) Registers are storage locations within the CPU itself.

⇒ True, registers store binary data using a collection of D flip-flops

4) A two-pass assembler generally creates a symbol table during the first pass and finishes the complete translation from assembly language to machine instructions on the second.

⇒ True

5) The MAR, MBR, PC, and IR registers in MARIE can be used to hold arbitrary data values.

⇒ False, only the AC is a general-purpose register

6) MARIE has a common bus scheme, which means a number of entities share the bus.

⇒ True, all registers and memory can access the bus but only one at a time

7) One-to-One correspondance b/w assembly language & machine instructions.

⇒ True

8) If a computer uses microprogrammed control, the microprogram determines the instruction set for the machine.

⇒ True