NAME: Steven Reeves

CST 131 HW#1

1. (10 Pts) Given the MARIE assembly language program below, provide the machine code in hex that is contained in the following addresses

Address	Machine code/Hex
101	3108
102	9106
105	7000
108	0023
10A	0021

	ORG 100			
[100]->	LOAD A	IR:	AC: 0023	PC: 0101
[101]->	ADD ONE	IR:	AC: 0024	PC: 0102
[102]->	JUMP S1	IR:	AC: 0024	PC: 0103
[103]->S2,	ADD ONE	IR:	AC: 0048	PC: 0104
[104]->	STORE A	IR:	AC: 0048	PC: 0105
[105]->	HALT	IR:	AC:	PC: 0106
[106]->S1,	ADD A	IR:	AC: 0047	PC: 0107
[107]->	JUMP S2	IR:	AC: 0047	PC: 0108
[108]->A,	HEX 0023	HEX 0048		
[109]->ONE,	HEX 0001			
[10A]->B,	DEC 33	33dec = (16^1*2) + (16^0*1) = 32 + 1		

b) What is the content in Hex of the memory address referenced by the label A after the execution of the HALT instruction?

Answer in Hex: 0048 (HEX 0023 + HEX 001 + HEX 0023 + HEX 001)

c) What is the memory location in Hex that is referenced to by the label A?

Answer in Hex: 108

2. (10 Pts) Using MARIE, provide the assembly language instruction when given the following machine language instructions in HEX:

Machine	Assembly
A000	Clear 000
000B	JnS 00B
C00F	JumpI 00F
9000	Jump 000

3. (10 Pts) A computer architecture similar to MARIE has an IR that looks like below.

Opcodes	Address of operand
19 15	14 0

a. How many possible opcodes are there?

$2^5 = 32$ possible opcodes

b. How many addresses in powers of 2 in memory are directly addressable by the address section of the IR?

$2^{15} = 32768$ directly addressable memory locations

c. What is the most negative decimal number that can be accommodated in one address of memory assuming the number of bits in an address is equal to the number of bits in the IR?

20 bits total, $2^{20} – 1 = -1048575$

5. (10 Pts) How many bits (lines) is required to address a 2M x 32 memory if

a. the memory is byte addressable

Horizontal lines = 2M

Vertical addressable places = 32bits /8bits (width/ addressable chunk)= 4

Total lines = Vertical + Horizontal = $2^1 + 2^{20} + 2^2 = 2^{23}$

2 x 2^20 (Mega) 32/8 (address for each byte in the line) = 2^1 * 2^20 * 2^2 = 2^ 23 = 8388608
23 bits

b. the memory is word addressable. (Hint: in the case, a word is 32 bits)

2 * 2^20 * 2 = 2^1 * 2^20 * 2^1 = 2^ 22 = 4194304 22 lines, or bits

4. (10 Pts) A certain memory space is said to be byte addressable with each word consisting of 4 bytes. There are 16 Mbytes total in this memory.

a. How many word addresses in this memory space?

Total lines / vertical addressable spaces = horizontal lines (word lines)
16 *2^20 / 2^2 (4 bytes per word) = 16 *2^18 = 4 * 2^2 *2^18 = 4 * 2^20 = 4MB or 4194304

b. Below, draw a block diagram of this memory space showing word addresses of the first 4 words and the last 2 words in this memory.

Word address	Byte 1	Byte 2	Byte 3	Byte 4
000				
001				
002				
003				
004				
…	…	…	…	…
FFE				
FFF				

5. (10 Pts) Describe in your own words the difference between a direct and indirect addressing mode.

Direct addressing just shows the opcode where to get information from. Indirect addressing shows the opcode where to get the address to get the information from. Indirect addressing is similar to using a pointer. It's as if direct addressing shows the mouse where the cheese is, and indirect shows the mouse where the directions to the cheese is.

6. (50 Pts) In the MARIE simulator, write and test a MARIE assembly program that employs a subroutine (i.e. utilize opcodes JnS and JumpI) to provide the product of two

numbers inputted from the keyboard. The same program will also employ another subroutine to provide the quotient and remainder of the same two numbers inputted from the keyboard. Label the two numbers X and Y. X is the first number inputted. For the quotient and remainder, use X/Y. This problem will be checked off in class.

9. Assume you have a byte-addressable machine that uses 32-bit integers and you are storing the hex value 1234 at address 0.

a. Show how this is stored on a big endian machine.
b. Show how this is stored on a little endian machine.
c. If you wanted to increase the hex value to 123456, which byte assignment would be more efficient, big or little endian? Explain your answer.

0 (Big Endian)			12	34

0 (Little Endian)			34	12

Little Endian would be more efficient. You only need to add '56' to one byte. Big Endian would have to move '12' and '34' before adding '56'.

0 (Little Endian)		56	34	12

10. Show how the following values would be stored by byte-addressable machines with 32-bit words, using little endian and then big endian format. Assume each value starts at address 10_{16}. Draw a diagram of memory for each, placing the appropriate values in the correct (and labeled) memory locations.

Base 16
a. 456789A1
b. 0000058A
c. 14148888
Hai

a)

Big Endian	00	01	10	11
10	45	67	89	A1

b)

Big Endian	00	01	10	11
10	00	00	05	8A

c)

Big Endian	00	01	10	11
10	14	14	88	88

a)

Little Endian	00	01	10	11
10	A1	89	67	45

b)

Little Endian	00	01	10	11
10	8A	05	00	00

c)

Little Endian	00	01	10	11
10	88	88	14	14

11. The first two bytes of a 2M x 16 main memory have the following hex values:
Byte 0 is FE
Byte 1 is 01
If these bytes hold a 16-bit two's complement integer, what is its actual decimal value if:
a. memory is big endian?
b. memory is little endian?
Bon

Big Endian
Byte 0 and Byte 1 = FE 01 = 1111 1110 0000 0001
Twos → 0000 0001 1111 1110 + 1 = 0000 0001 1111 1111 = 256 + 255 = -511
Twos → <flip bits> <add one> = <convert this> = 256 + 255 = <answer is negative>

Little Endian
Byte 0 and Byte 1 = FE 01 = 0000 0001 1111 1110
Twos → 1111 1110 0000 0001 +1 = 1111 1110 0000 0010 = 65026