gabe malone

CSCI210 Assembly Language and Computer Systems
Lab Assignment

TASK One:

Pretend that the following table is an array of memory cells, each 1 byte in size. The base address of this block is 0x00

Column One => Base address of row (0x00, 0x0A, 0x15)

Row One => Offset

You can calculate the effective address by ***base + offset***.

Example: the effective address of the last cell of the first row is **0x00 + 0x09 = 0x09**

Example: the effective address of the third cell in the second row is **0x0A + 0x2 = 0x0C**

BASE	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
0x00	AD	DE	FE	CA	EF	BE	50	51	73	Ex 1 37
0x0A	55	CA	Ex 2							
0x14										

Given the following values show how the bytes would be **LITTLE ENDIAN** ordered in memory by filling the values into the table above beginning with address 0x00[0].

NOTE: Fill these data in one after another in the above table. Do not worry if the table is not completely filled. Just like RAM, there will be some available memory

1. 16 bit HALF WORD varOne => 0xDEAD

2. 32 bit WORD varTwo => 0xBEEFCAFE

3. 16 bit HALF WORD varThree => 0x5150

4. 32 bit WORD varFour => 0xCA553773

Based on the filled in table above, answer the following questions

1. What is the effective address of the **low order byte** of varTwo?

   0x02

2. What is the effective address of the **high order byte** of varFour?

   0x0B

3. What is the effective address of the **high order byte** of varThree

   0x07

BASE	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
0x00	B0	A7	DE	AD	BE	A7	51	50	00	DE
0x0A	FE	C8								
0x14										

Given the following values show how the bytes would be **BIG ENDIAN** ordered in memory by filling the values into the table above beginning with address 0x00[0]

1. 16 bit HALF WORD varOne => 0xBOA7
2. 32 bit WORD varTwo => 0xDEADBEA7
3. 16 bit HALF WORD varThree => 0x5150
4. 32 bit HALF WORD varFour => 0x00DEFEC8

Based on the filled in table above, answer the following questions

1. What is the effective address of **the low order byte** of varTwo?

   0x05

2. What is the effective address of the **high order byte** of varFour?

   0x08

3. What is the effective address of the **high order byte** of varThree

   0x06

Consider the following binary data starting at address 0xFFFFF1A

1A 1B 1C 1D 1E 1F 20

01000011 43	01001111 4F	01001101 4D	01010000 50	00100000 20	00110001 31	00110000 30
00110101 35	~~00100001~~ 21					

21 22

1. If this data represents a ***byte array*** of 9 numbers, what are the numbers in the array in decimal notation? What are the addresses of the 9 numbers?

   Addresses: 0xFFFFF1A, ...1B, ...1C, ...1D, ...1E, ...1F, ...20, ...21

   $values_{10}$ → 67 79 77 80 32 49 48 53

2. If they represent a ***little endian 32-bit word array*** of 4 numbers, what are the numbers in the array in decimal notation? What are the addresses of the ~~4~~ 2 numbers?

   Addresses: 1A, ... 1E → 35 30 31 20 → 892,350,752

   Value → 50 4D 4F 43 → 1,347,243,843

3. If they represent a ***big endian 32-bit word array*** of 4 numbers, what are the numbers in the array in decimal notation?

   1A → 43 4F 4D 50 → 1,129,270,608

   1E → 20 31 30 35 21 → 138,264,196,385

Consider the following binary data starting at address 0xFFFFF1A

01000011 C	01001111 O	01001101 m	01010000 P	00100000 ␣	00110001 1	00110000 0
00110101 5	00100001 !					

1. If the 9-byte binary string starting at **0xFFFFF1A** represents an ASCII string, what string do they represent?

   Comp 105!

2. What is the hexadecimal ASCII representation for the string "Metal"?

   4D 65 74 61 6C

3. What is the binary ASCII representation for the string "Metal"?

   1100

   0100 1101 0110 0101 0111 0100 0110 0001 0110

4. What is the hexadecimal representation for the string "Cows"?

   43 6F 77 73

5. What is the hexadecimal representation for the string "COWS"?

   43 4F 57 53

Booths Algorithm: Assuming 8 bits per value and a 16-bit destination register, provide an algorithm trace of Booth's Algorithm using the following expression.

0111

- 7 x -3

asb = 0011
1100

1100
+ 1
1101

(7) 0111

(-3) 1101

Product: 0000 0111

- Fill in the table with the values of the following for each iteration
    - The iteration number
    - The high order byte of destination register
    - The low order byte of destination register
    - The current bit
    - The previous bit
    - The operation that will be performed

- Things to consider:
    - If you were writing a software implementation of this algorithm (which you will do) how would you accomplish the following? (answer these questions in painful detail)
        - Store and access the previous bit
        - Test the current bit
        - Perform a mathematical operation on the high order half of a word

Iteration	HO	LO	CB	PB	Operation (nothing, add , subtract)
1	0000	0111	1	0	subtraction, >>
2	0001	1011	1	1	nothing >>
3	0000	1101	1	1	nothing >>
4	0000	0110	0	1	add >>
	1110	1011			done
	2's complement to check				0001 - 0100 + 1 = 0101
					-21