# EE 215  Homework #2

(5 problems, 104 pts total)

1. 2's complement math (35 pts total, 5 pts each)

   Two's complement is the most commonly used format for signed numbers (positive and negative) in digital systems.

   Using 8 bits, show:

   (a) What is the maximum and minimum number that can be represented?

   (b) $9_{10} + 80_{10}$

   (c) $120_{10} + 10_{10}$

   (d) $5_{10} - 12_{10}$

   (e) $10_{10} - 140_{10}$

   (f) Did an error occur in any of these operations? How can you tell?

   The MSP430 uses 16 bits for representing numbers.

   (g) What is the maximum and minimum number that can be represented using 2's complement with 16 bits?

***Answers:***

(a)

In the complement, the first digit represents a sign, 0 represents a positive number, and 1 represents a negative number. Therefore:

The maximum number:$0111\ 1111_2=0*2^7+1*2^6+1*2^5+1*2^4+1*2^3+1*2^2+1*2^1+1*2^0=127_{10}$

The minimum number:$1000\ 0000_2=-(0111\ 1111_2)-1=-128_{10}$

(b)

  0000 1001

$\underline{+0101\ 0000}$

  0101 1001=$89_{10}$

(c)

+0111 1000

+0000 1010

+1000 0010=$-126_{10}$

At eight bits, two positive numbers add up to a negative number, which means an overflow. Eight bits can only count up to plus 127.

(d)

```
  0000 0101
+1111 0100
  1111 1001=-7₁₀
```

(e)

At eight bits, it cannot represent -140, so this formula cannot be calculated.

(f)

Question (c) and question (e) have errors.

An error occurs when a number exceeds the upper limit that eight bits can represent, such as when two positive numbers add up to a negative number, or when two negative numbers add up to a positive number.

2. Register operations (15 pts total, 5 pts each)

   See “MSP430 Microcontroller Basics”, section 5.1
   Or “MSP430 x5xx Family User’s Guide” section 6.3

   (a) How many registers are in the MSP430?

   (b) What are the special registers: PC, SP, SR ? Write one sentence to describe them.

   See “MSP430 x5xx Family User’s Guide” section 6.5 and 6.6 for full listing of assembly language commands.

   (c) List five of the assembly language commands, and tell what they do.

***Answers:***

(a) 16 registers.

(b)

PC (Program Counter): It holds the memory address of the next instruction to be fetched and executed by the CPU, effectively pointing to the current instruction in a program.

SP (Stack Pointer): This register keeps track of the top address of the stack in memory, facilitating the management of function calls and data storage in a stack-based architecture.

SR (Status Register): Also known as the flag register, it stores various condition flags that reflect the outcome of operations, such as whether a result is zero, negative, or if there was a carry or overflow in arithmetic operations, aiding in program flow control.

(c)

(1) INC : Add the operand by one.

(2) ADD : Add the two operands and save the result in the destination operand.

(3) DEC : Subtract the operand by one.

(4) MOV : Assigns the value of the current operand to the destination operand.

(5) NOP : Do nothing.

3. ALU control signals. (10 pts total, 2 pts each)

For the following control signals, $C_2C_1$ and $C_5C_4$ are used to select source register R[0]-R[3], $C_3$ and $C_6$ is used for $D_{in}$ selection. If $C_3 = 1$, MUX1 output is $D_{in}$. If $C_6 = 1$, MUX2 output is $D_{in}$. $C_8, C_7$ are used for the 2 bit binary decoder to select destination register. $C_{11}C_{10}C_9$ are used for ALU selection, and the details are listed in the following table.

| Control signal | 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 |
|---|---|---|---|---|---|---|---|---|
| Operation | A and B | A+B | A^B | A-B | ~(AB) | ~(A+B) | A*B | A/B |

(a) If $C_{11}C_{10}C_9C_8C_7C_6C_5C_4C_3C_2C_1 = 11011010001$, describe the corresponding register transfer and ALU operation.

(b) If $C_{11}C_{10}C_9C_8C_7C_6C_5C_4C_3C_2C_1 = 00100011001$, describe the corresponding register transfer and ALU operation.

(c) To accomplish $R[3] \leftarrow D_{in} - R[0]$, what control signals should be used?

(d) To accomplish $R[2] \leftarrow R[2]/R[0]$, what control signals should be used?

(e) To accomplish $R[0] \leftarrow R[1]^3$, how many steps do we need? What are the control signals for each step?

***Answers:***

(a)

R[3] ← R[1] * R[2]

(b)

R[0] ← R[1] + R[3]

(c)

$C_{11}C_{10}C_9C_8C_7C_6C_5C_4C_3C_2C_1$ =011 11 000 1XX

(d)

$C_{11}C_{10}C_9C_8C_7C_6C_5C_4C_3C_2C_1$ =111 10 000 010

(e)

2 steps

(1)

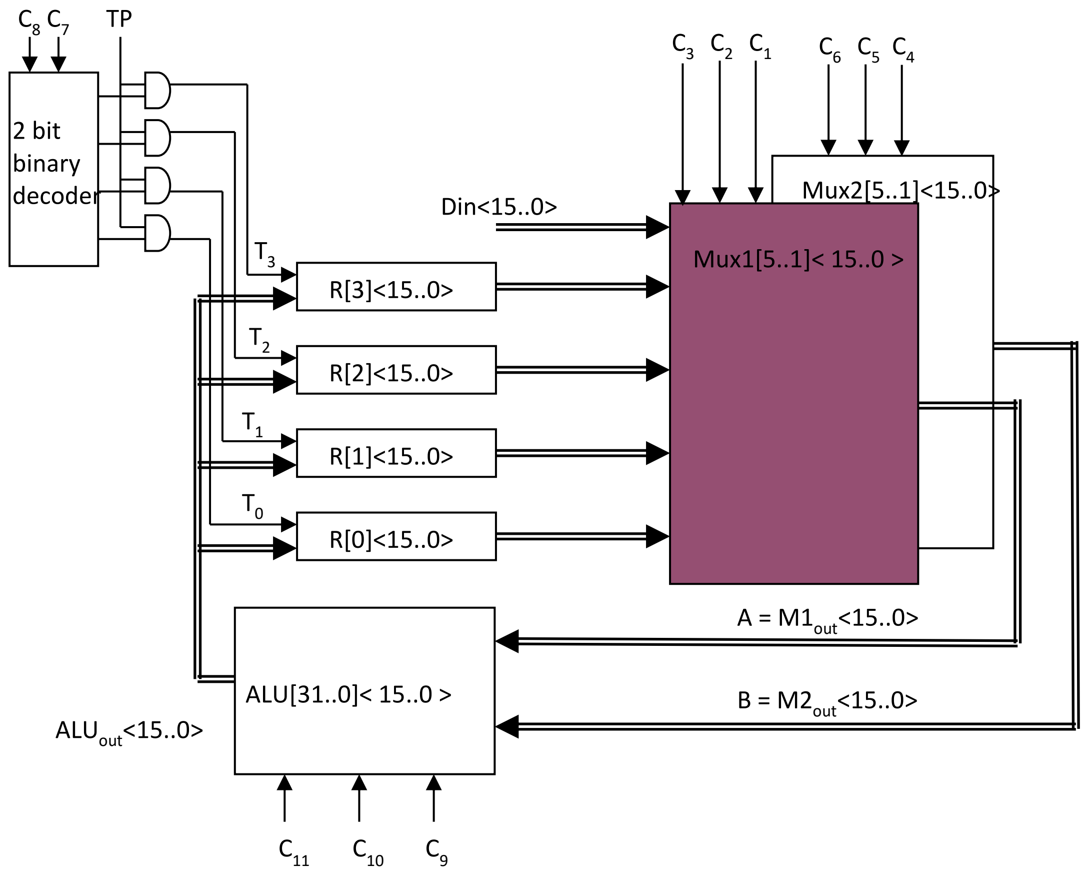
$R[0] \leftarrow R[1] * R[1]$

$C_{11}C_{10}C_9C_8C_7C_6C_5C_4C_3C_2C_1$ =110 00 001 001

(2)

$R[0] \leftarrow R[0] * R[1]$

$C_{11}C_{10}C_9C_8C_7C_6C_5C_4C_3C_2C_1$ =110 00 000 001



4. Assembly language (24 pts total, 2 pts each)

For each assembly language instruction, describe what is happening by a drawing of the register contents before and after. (Assume the start before each command is R4 = 0x4200, R5=0x022A, R6 = 0x7736) [see MSP430 x5xx Family Users Guide section 6.6.2]

```
(a)  mov.w  R4,R6
(b)  add.w  R4,R5
(c)  and.b  R4,R6
(d)  bis.b  #0011b,R6
(e)  clr.w  R5
(f)  dec.b  R6
(g)  inc.w  R4
(h)  inv.b  R6
(i)  rla.w  R6
(j)  sub.w  R5,R4
(k)  swpb   R6
(l)  xor    R4,R5
```

***Answers:***

`(a) mov.w R4,R6`

Assign the value of R4 to R6

before: R4= 0x4200, R6 = 0x7736

after: R6 = 0x4200

`(b) add.w R4,R5`

Add R4 to R5 and save the results in R5

before: R4=0x4200 R5=0x022A

after: R5 = 0x442A

`(c) and.b R4,R6`

Perform logical AND for each bit in R4 and R6, respectively

before: R4= 0x4200, R6 = 0x7736

R4: 0000 0000

and with R6: $\underline{0011\ 0110}$

0000 0000 $=00_{16}$

after: R6= 0x0000

`(d) bis.b #0011b,R6`

Perform logical OR for each bit, respectively

0000 0011

OR R6:<u>0011 0110</u>

0011 0111 =$37_{16}$

after: R6= 0x0037

`(e) clr.w R5`

Clear the value in the R5 register

before: R5= 0x022A

after: R5= 0x0000

`(f) dec.b R6`

Subtract the value in R6 by one

before: R6 = 0x7736

after: R6 = 0x0035

`(g) inc.w R4`

Add one to the value in the R4 register

before: R4 = 0x4200

after: R4 = 0x4201

`(h) inv.b R6`

Inverts the value at each position in the operand

before: R6 = 0x7736

R6: 0011 0110

invert R6: 1100 1001=$C9_{16}$

after: R6 = 0x00C9

`(i) rla.w R6`

All bits are shifted one bit to the left, with the lowest bit filled with 0

before: R6 = 0x 7736

R6: 0111 0111 0011 0110

after: 1110 1110 0110 1100 =$EE6C_{16}$

after: R6 = 0x EE6C

`(j) sub.w R5,R4`

The target number minus the operand, means R4-R5.

before: R4= 0x4200, R5= 0x022A

R4=$16896_{10}$ R5=$554_{10}$

So R4-R5=16896-554=16342=0x4177

after:, R4=0x3FD6 R5 = 0x022A

`(k) swpb R6`

Swap the first half with the second half

before: R6 = 0x 7736

after: R6 = 0x 3677

`(l) xor R4,R5`

Perform an exclusive OR between the operand and destination

before: R4 = 0x4200 = 0100 0010 0000 0000

R5 = 0x022A = 0000 0010 0010 1010

XOR: 0100 0000 0010 1010 =$402A_{16}$

after: R5 = 0x 402A

5. New words. (20 pts total, 5 pts each).

   Define these words in one or two sentences.

(a) ALU

(b) fetch/execute cycle

(c) word (what data length is this?)

(d) MAR

(e) microcode

***Answers:***

a. ALU (Arithmetic Logic Unit): It's the part of the CPU responsible for performing arithmetic and logic operations on data, such as addition, subtraction, and comparisons. It has two inputs and one output. The ALU also provides signals C, N, V, Z which tells register the information about the output value.

b. Fetch/Execute Cycle: The fundamental operation of a CPU where it fetches an instruction from memory, decodes it, executes it, and then repeats the process for the next instruction in a program.

c. Word: In the context of assembly language, a "word" typically refers to a fixed-sized unit of data, often representing the natural word size of the CPU (e.g., 16 bits = 2 bytes. a word is stored in 2 bytes of memory because the memory has addresses for every byte).

d. MAR (Memory Address Register): It's a register that holds the memory address of the data or instruction being accessed or manipulated in memory.

e. Microcode: Microcode is a low-level control code used internally by a CPU to execute machine-level instructions. It interprets and carries out the instructions fetched from memory.