EE 215 Homework #2 Solution

(5 problems, 104 pts total)

Instructions: Write your name, class number, student number, HW2, and date on top of page. **Box all answers**. Due at beginning of class. **Staple upper left** if you have more than one page. Show all work to get credit.

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?

Solution:

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

The first bit indicates the sign, 0 is positive, 1 is negative

Max: $0111\ 1111_2 = \text{0x7F} = 127_{10}$

Min: $1000\ 0000_2 = \text{0x80} = -128_{10}$

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

$$\begin{array}{l} 0000\ 1001 = \text{0x09} = 9_{10} \\ \underline{+0101\ 0000\ =}\ \text{0x50} = 80_{10} \\ 0101\ 1001 = \text{0x59} = 89_{10} \end{array}$$

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

0111 1000 = 0x78 = 120_{10}

+0000 1010 = 0x0A = 10_{10}

1000 0010 = 0x82 = -126_{10}

This must be a mistake, cannot add two positive numbers and get a negative!

This is the overflow condition, and would set the V flag.

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

0000 0101 = 0x05 = 5_{10}

+1111 0100 = 0xF4 = -12_{10}

1111 1001 = 0xF9 = -7_{10}

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

0000 1010 = 0x0A = 10_{10}

The number -140 cannot be represented, so the computer cannot do this calculation with only 8 bits.

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

An error occurs if you cannot represent the number, if you overflow (the number is too small or too big to represent with the given number of bits).
(1) you add two negative numbers and get a positive number that is overflow.
(2) you add two positive numbers and get a negative number, that is an overflow.

Part (c) and (e) have errors.

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

+32767, and -32768

Max: *$0111\ 1111\ 1111\ 1111_2$ = 0x7FFF = 32767_{10}*

Min: *$1000\ 0000\ 0000\ 0000_2$ = 0x8000 = -32768_{10}*

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.

Solution:

(a) MSP430 has 16 registers.

(b) What are the special registers: PC, SP, SR?

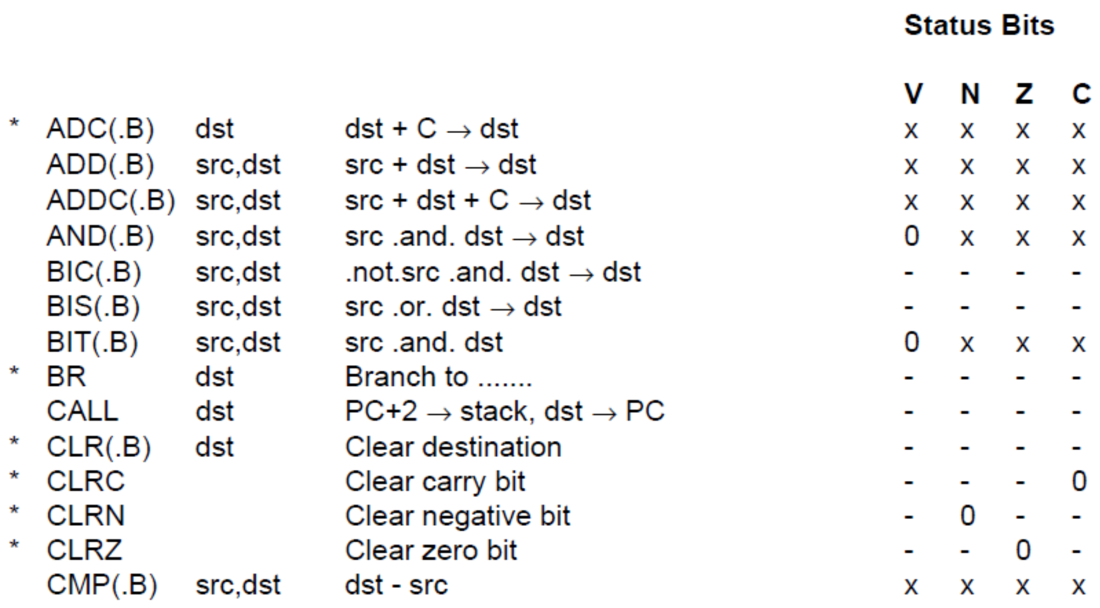
PC = program counter. This register contains the address of the next instruction to be executed. Each instruction is 1-3 words of memory, always on even addresses.

SP = stack pointer. The stack is a last-in-first-out (LIFO) data structure. The stack pointer is a register, and contains (points to) an address in memory that is the top of the stack.

SR = status register. The status register has a set of single bits (flags). The C, Z, N, and V bits are affected by the output operation of the ALU. Four other bits control the mode of operation of the microcontroller for low power. One other bit is a general interrupt enable.

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

Any five of the list that follows:



				Status Bits			
				V	**N**	**Z**	**C**
*	ADC(.B)	dst	dst + C → dst	x	x	x	x
	ADD(.B)	src,dst	src + dst → dst	x	x	x	x
	ADDC(.B)	src,dst	src + dst + C → dst	x	x	x	x
	AND(.B)	src,dst	src .and. dst → dst	0	x	x	x
	BIC(.B)	src,dst	.not.src .and. dst → dst	-	-	-	-
	BIS(.B)	src,dst	src .or. dst → dst	-	-	-	-
	BIT(.B)	src,dst	src .and. dst	0	x	x	x
*	BR	dst	Branch to	-	-	-	-
	CALL	dst	PC+2 → stack, dst → PC	-	-	-	-
*	CLR(.B)	dst	Clear destination	-	-	-	-
*	CLRC		Clear carry bit	-	-	-	0
*	CLRN		Clear negative bit	-	0	-	-
*	CLRZ		Clear zero bit	-	-	0	-
	CMP(.B)	src,dst	dst - src	x	x	x	x

				V	N	Z	C
				Status Bits			
*	DADC(.B)	dst	dst + C → dst (decimal)	x	x	x	x
	DADD(.B)	src,dst	src + dst + C → dst (decimal)	x	x	x	x
*	DEC(.B)	dst	dst - 1 → dst	x	x	x	x
*	DECD(.B)	dst	dst - 2 → dst	x	x	x	x
*	DINT		Disable interrupt	-	-	-	-
*	EINT		Enable interrupt	-	-	-	-
*	INC(.B)	dst	Increment destination, dst +1 → dst	x	x	x	x
*	INCD(.B)	dst	Double-Increment destination, dst+2→dst	x	x	x	x
*	INV(.B)	dst	Invert destination	x	x	x	x
	JC/JHS	Label	Jump to Label if Carry-bit is set	-	-	-	-
	JEQ/JZ	Label	Jump to Label if Zero-bit is set	-	-	-	-
	JGE	Label	Jump to Label if (N .XOR. V) = 0	-	-	-	-
	JL	Label	Jump to Label if (N .XOR. V) = 1	-	-	-	-
	JMP	Label	Jump to Label unconditionally	-	-	-	-
	JN	Label	Jump to Label if Negative-bit is set	-	-	-	-
	JNC/JLO	Label	Jump to Label if Carry-bit is reset	-	-	-	-
	JNE/JNZ	Label	Jump to Label if Zero-bit is reset	-	-	-	-
	MOV(.B)	src,dst	src → dst	-	-	-	-
*	NOP		No operation	-	-	-	-
*	POP(.B)	dst	Item from stack, SP+2 → SP	-	-	-	-
	PUSH(.B)	src	SP - 2 → SP, src → @SP	-	-	-	-
	RETI		Return from interrupt TOS → SR, SP + 2 → SP TOS → PC, SP + 2 → SZP	x	x	x	x
*	RET		Return from subroutine TOS → PC, SP + 2 → SP	-	-	-	-
*	RLA(.B)	dst	Rotate left arithmetically	x	x	x	x
*	RLC(.B)	dst	Rotate left through carry	x	x	x	x
	RRA(.B)	dst	MSB → MSB ....LSB → C	0	x	x	x
	RRC(.B)	dst	C → MSB ....LSB → C	x	x	x	x
*	SBC(.B)	dst	Subtract carry from destination	x	x	x	x
*	SETC		Set carry bit	-	-	-	1
*	SETN		Set negative bit	-	1	-	-
*	SETZ		Set zero bit	-	-	1	-
	SUB(.B)	src,dst	dst + .not.src + 1 → dst	x	x	x	x
	SUBC(.B)	src,dst	dst + .not.src + C → dst	x	x	x	x
	SWPB	dst	swap bytes	-	-	-	-
	SXT	dst	Bit7 → Bit8 .... Bit15	0	x	x	x
*	TST(.B)	dst	Test destination	x	x	x	x
	XOR(.B)	src,dst	src .xor. dst → dst	x	x	x	x

Legend:				
	0	The Status Bit is cleared	1	The Status Bit is set
	x	The Status Bit is affected	-	The Status Bit is not affected
	*	Emulated Instructions		

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

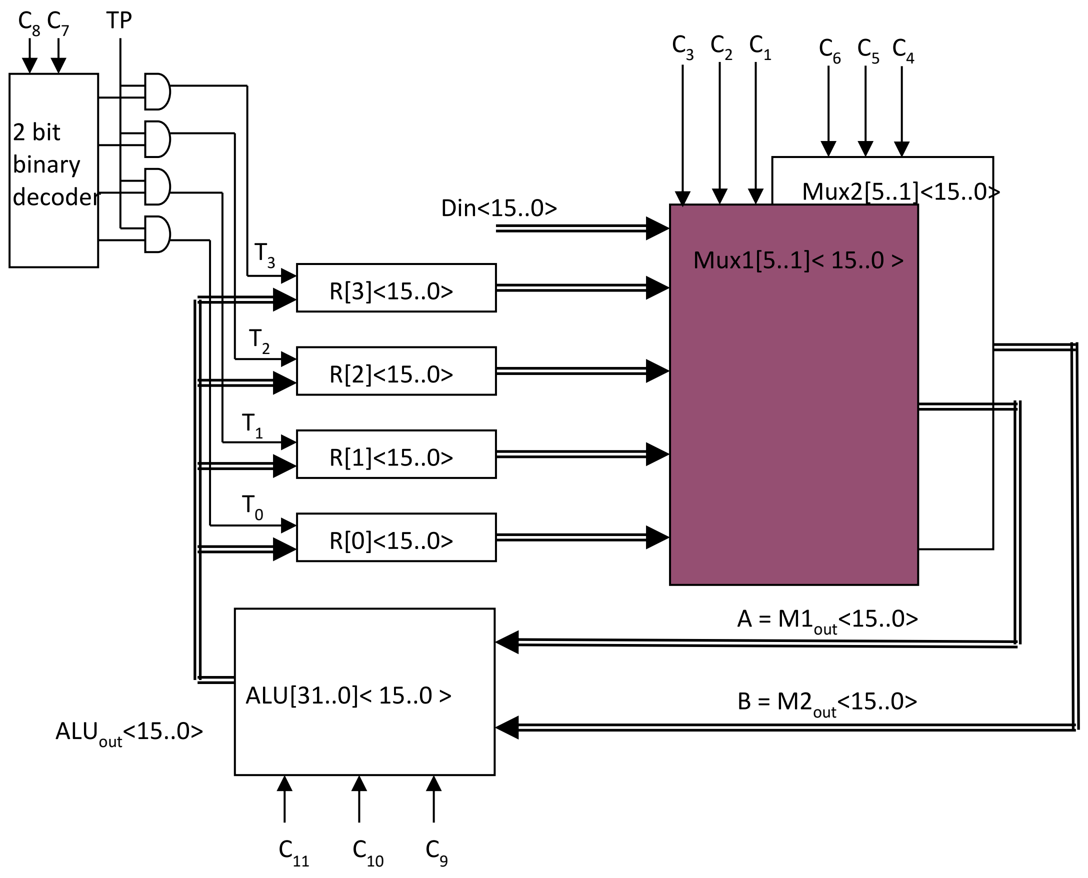
(a) If $C_{11}C_{10}C_9\ \ C_8C_7\ \ C_6C_5C_4\ \ C_3C_2C_1 = 110\ \ 11\ \ 010\ \ 001$, describe the corresponding register transfer and ALU operation.

(b) If $C_{11}C_{10}C_9\ \ C_8C_7\ \ C_6C_5C_4\ \ C_3C_2C_1 = 001\ \ 00\ \ 011\ \ 001$, 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?



Solution:

(a) If $C_{11}C_{10}C_9\ \ C_8C_7\ \ C_6C_5C_4\ \ C_3C_2C_1 = 110\ \ 11\ \ 010\ \ 001$, describe the corresponding register transfer and ALU operation.

$$R[3] \leftarrow R[1] * R[2]$$

(b) If $C_{11}C_{10}C_9\ \ C_8C_7\ \ C_6C_5C_4\ \ C_3C_2C_1 = 001\ \ 00\ \ 011\ \ 001$, describe the corresponding register transfer and ALU operation.

$$R[0] \leftarrow R[1] + R[3]$$

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

$$C_{11}C_{10}C_9\ \ C_8C_7\ \ C_6C_5C_4\ \ C_3C_2C_1 = 011\ \ 11\ \ 000\ \ 1XX$$

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

$$C_{11}C_{10}C_9\ \ C_8C_7\ \ C_6C_5C_4\ \ C_3C_2C_1 = 111\ \ 10\ \ 000\ \ 010$$

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

Two steps:

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

$$C_{11}C_{10}C_9\ \ C_8C_7\ \ C_6C_5C_4\ \ C_3C_2C_1 = 110\ \ 00\ \ 001\ \ 001$$

(2) $R[0] \leftarrow R[0] * R[1]$

$$C_{11}C_{10}C_9\ \ C_8C_7\ \ C_6C_5C_4\ \ C_3C_2C_1 = 110\ \ 00\ \ 000\ \ 001$$

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
```

Solution:

```
(a) mov.w  R4,R6
                copies contents of R4 to R6
                before: R4= 0x4200, R6 = 0x7736
                after:  R6 = 0x4200

(b) add.w  R4,R5
                adds the contents of R4 to R5 and puts result in R5
                before:  R4=0x4200  R5=0x022A
                after:  R5 = 0x442A

(c) and.b  R4,R6
                     takes the logical AND of each bit
                     before:  R4= 0x4200, R6 = 0x7736
                     low byte of R4: 0000 0000
                     and with R6   : 0011 0110
                     result:         0000 0000
                     in hex is:         0    0
                     after:  R6= 0x0000

(d) bis.b  #0011b,R6
                     sets the bits by using the OR function
                     first operand:  0000 0011
                     OR R6:          0011 0110
                     result:         0011 0111
                     in hex is:        3    7
                     after:  R6= 0x0037

(e) clr.w  R5
                     clears the contents of the register
                     before:  R5= 0x022A
                     after:   R5= 0x0000

(f) dec.b  R6
                     decrement will decrease the value by 1
                     before:  R6 = 0x7736
                     after:   R6 = 0x0035

(g) inc.w  R4
                     increment will increase the operand by 1
                     before:  R4 = 0x4200
                     after:   R4 = 0x4201

(h) inv.b  R6
                     inverts every bit in the operand
                     before:  R6 = 0x7736
                     low byte of R6: 0011 0110
```

```
                 inverted:       1100 1001
                 in hex this is:  C    9
                 after:   R6 = 0x00C9

(i) rla.w  R6
                 rotate left arithmetically (signed multiplication by 2)
                 each bit is shifted left, and the LSb is filled with 0.
                 before:  R6 = 0x 7736
                 each bit:  0111 0111 0011 0110
                 after:     1110 1110 0110 1100
                 in hex:      E    E    6    C
                 after:   R6 = 0x EE6C

(j) sub.w  R5,R4
                 subtract source from destination, so this is R4-R5
                 before: R4= 0x4200, R5= 0x022A
                 can convert these to decimal to check:
                 16896-554=16342, so that is 0x4117
                 after:  R5 = 0x022A, R4=0x3FD6

(k) swpb   R6
                 this is swap bytes.
                 before: R6 = 0x 7736
                 after:  R6 = 0x 3677

(l) xor    R4,R5
                 The Exclusive OR of the source and destination
                 before:
                 R4 = 0x4200 = 0100 0010 0000 0000
                 R5 = 0x022A = 0000 0010 0010 1010
                 XOR:          0100 0000 0010 1010
                 result in hex:  4   0    2    A
                 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

Solution:

(a) ALU

An arithmetic logic unit (ALU) is a digital circuit used to perform arithmetic and logic operations. It represents the fundamental building block of the central processing unit (CPU) of a computer.
It has two inputs and one output. The ALU also provides signals C, N, V, Z to the status register to tell information about the output value.

(b) fetch/execute cycle

The fetch execute cycle is the basic operation (instruction) cycle of a computer (also known as the fetch decode execute cycle).
It is a cycle of getting the next instruction from memory, decoding the instruction, executing the instruction, storing the result, incrementing the PC and back to the cycle. (c) word (what data length is this?)

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

A word is 16 bits = 2 bytes, most data in memory is a word.
The memory has addresses for every byte, so a word is actually stored in 2 bytes of memory.

(d) MAR

In a computer, the Memory Address Register is the CPU register that either stores the memory address from which data will be fetched from the CPU, or the address to which data will be sent and stored.
MAR keeps track of the location in memory.

(e) microcode

Microcode is a layer of hardware-level instructions which are stored permanently in a computer or peripheral controller and controls the operation of the device.
Microcode implements higher-level machine code instructions or internal state machine sequencing in many digital processing elements.