Department of Electrical and Computer Engineering
University of Alabama in Huntsville

CPE 323 – Introduction to Embedded Computer Systems

Midterm Exam Keys

Instructor: Dr. Aleksandar Milenkovic

Date: October 07, 2015

Place: EB 207

Time: 2:20 PM – 03:40 PM

Note: Work should be performed systematically and neatly. This exam is closed books and closed neighbour(s). Allowable items include exam, pencils, straight edge, calculator, and materials distributed by the instructor. Good luck!

Question	Points	Score
1	15	
2	25	
3	20	
4	20	
5	20+5	
Sum	100+5	

Please print in capitals:

Last name:_____

First name: _____

1. (15 points) Misc, MSP430

Circle the correct answer for A-E and type in the answers for F and G.

1.A. (True | False) (2 points) Assembly language directive "DC16 4" allocates 8 bytes in memory and initializes their values to 0x00.

1.B. (True | False) (2 points) Assembly language directive "DS32 8" allocates 32 bytes in memory.

1.C. (True | False) (2 points) Register R1 serves as the stack pointer (SP).

1.D. (True | False) (2 points) Flags C, V, Z, and N residing in the status register (R2) can be modified by software developers using assembly language instructions (e.g., BIC and BIS instructions).

1.E. (True | False) (2 points) A push onto the program stack will increase the value of the stack pointer.

1.F. (2 points) How many memory operations (reads from memory and writes to memory) occurs during execution of the instruction MOV.W #4523, &WDTCTL?

3 to fetch the instruction and 1 to write the result => 4 memory operations

1.G. (3 points) What is the address range of a 4 KB block of data placed in memory at the address 0x0C00? Fill in the blanks.

[0x0C00__ – 0x1BFF]

2. (25 points) Assembler (Directives, Instructions, Addressing Modes)

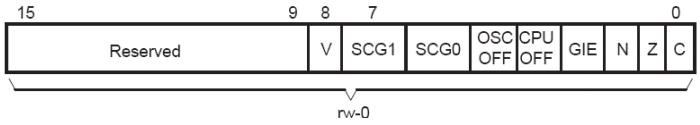
2.A. (5 points) Show the word-wide **HEXADECIMAL** content of memory corresponding to the following sequence of assembler directives. ASCII code for character 'A' is 65 decimal / 41 hex, and for character '0' is 48 decimal / 30 hex. Note: the number of rows does not reflect the number of words allocated by the directives.

```
// suffix q stands for octal, suffix b stands for binary, prefix 0x for hex
          ORG 0xE000
C1        DC8 0137q, 0xC8, 11000011b
          EVEN
C2        DC8 'A', '0', "BC"
          EVEN
C3        DC32 -6
```

Label	Address [hex]	Memory[15:0] [hex]
C1	0xE000	0xC85F
	0xE002	0x??C3
C2	0xE004	0x3041
	0xE006	0x4342
	0xE008	0x??00
C3	0xE00A	0xFFFA
	0xE00C	0xFFFF

2.B. (20 points) Consider the following instructions given in the table below. For each instruction determine addressing modes of the source and destination operands, source and destination addresses, and the result of the operation. Fill in the empty cells in the table. The initial content of memory is given in the table. The initial value of registers R2, R5, and R6 is as follows: SR=R2=0x0001 (V=0, N=0, Z=0, C=1), R5=0xC003, R6=0xC006. Assume the starting conditions are the same for each question, i.e., always start from the initial conditions in memory and given register values.

Label	Address [hex]	Memory[15:0] [hex]
	0xC000	0x0504
	0xC002	0xFEEE
TONI	0xC004	0x9862
	0xC006	0x3344
	0xC008	0xF014
DEN	0xC00A	0x2244
EDE	0xC00C	0xCDDA
	0xC00E	0xEFDD



	Instruction	**Instr. Size in Words**	**Source Operand Addressing Mode**	**Destination Operand Addressing Mode**	**Source Address**	**Dest. Address**	**Result (content of a memory location or a destination register;**
(a)	DADD.W @R6+, TONI	2	Register Indirect with Autoincrement	Symbolic	C006	C004	word operation, decimal addition: 0x9862 + 0x3344 +1(C) = 0x3207 => write the word to location C004 => M[C004] = 0x3207
(b)	SUBC &DEN, 8(R6)	3	Absolute	Indexed	0xC00A	C00E	word operation (dst + not. src + C) src = M[0xC00A] = 0x2244 dst = M[0xC00E] = 0xEFDD => result: EFDD+DDBB+1 = 0xCD99 => M[C00E] = 0xCD99
(c)	XOR.W @R6, R5	1	Register indirect	Register direct	C006	-	Src .xor. Dst src = 0x3344 dst = 0xC003 R5 = 0xF347

3. (20 points) Analyze assembly program

Consider the following code segment.

```
01        SUB     #4, SP    ; allocate 4 bytes (2 words on the stack)
02        MOV     mylw, 0(SP)           ;
03        MOV     mylw+2, 2(SP)         ;
04        BIT     #0x8000, 2(SP)        ;
05        JZ      lskip                 ;
06        INV     2(SP)                 ;
07        INV     0(SP)
08        ADD     #1, 0(SP)
09        ADDC    #0, 2(SP)
10 lskip:NOP
11
12 mylw DC32 0xFFFFFFFA
```

3.A. (2 points) How many bytes is allocated by the assembly directive in line 12?
4 bytes (0xFFFFFFFA).

3.B. (2 points) What is the content of register SP after the instruction in line 01 is completed? The initial value of SP is 0x1200.
Register SP=0x11FC.

3.C. (3 points) What is the content of memory locations at addresses [SP+0] and [SP+2] after the instructions in lines 02 and 03 are completed, respectively?
M[SP+0] = 0xFFFA;
M[SP+2] = 0xFFFF;

3.D. (3 points) What is the content of memory locations at addresses [SP+0] and [SP+2] after the program is executed (line 10)?
M[SP+0] = 0x0006;
M[SP+2] = 0x0000;

3.E. (8 points) What does this code segment do? Explain your answer.

This code segment places the absolute value of the long integer residing at the location with label mylw onto the stack. First, we push the original value which includes 2 words (0xFFFA and 0xFFFF). The sign bit of the operand (bit 32) is checked. If it is set, the 32-bit number is a negative and its first complement is found and then added 1 to get the 2's complement of the original number.

3.F. (2 points) Calculate the total execution time in seconds for the code sequence from above (line 01 – line 10). We know the following: the average CPI is 2 clocks per instruction. Assume the clock frequency is 1 MHz. What is MIPS rate for this code?
IC = 10
*ExTime = IC*CPI/ClockFreq = 10*2/1*10^6= 20 us*
*MIPS = IC/ExTime*10^6 = 0.5 MIPS*

4. (20 points, C language) Consider the following C program. Assume that the register SP at the beginning points to 0x0A00. Answer the following questions. Assume all variables are allocated on the stack, and in the order as they appear in the program. ASCII code for character '0' is 48 (0x30).

```
1   int main( void )  {
2     volatile int a = 4;
3     volatile long int c = -2, d = 2;
4     volatile char mych = {'4', '3', '2', '1'};
5     volatile long int *pli = &c;
6     Volatile int *pi = &a;
7     pli = pli - 1;    // pli = pli - 1*sizeof(long int)
8     pi = pi - 6;    //  pi = pi - 6*sizeof(int)
9     *pi = a + *pi;
10  }
```

Fill in the following table by determining the values/addresses given below.

#	Question?	Value/Address
1	The number of bytes allocated on the stack for the variable declared in line 2.	2 bytes
2	The number of bytes allocated on the stack for the character array declared in line 4.	4 bytes
3	The number of bytes allocated on the stack for all variables declared in lines 2-6.	18 bytes
4	Value of mych[0] after initialization performed in line 4.	'0'/ 0x34
5	Address of variable a (&a).	0x09FE
7	Value of pli at the moment after the statement in line 5 is executed.	0x09FA
8	Value of pli at the moment after the statement in line 7 is executed.	0x09F6
9	Value of pi at the moment after the statement in line 8 is executed.	0x09F2
10	Value of mych[0] at the moment after the statement in line 9 is executed.	'8'

0x0A00	TOS	Comments
0x09FE	0002	a
0x09FC 2	FFFF	c.upper
0x09FA 4	FFFE	c.lower
0x09F8 6	0000	d.upper
0x09F6 8	0002	d.lower
0x09F4 10	3132	mych[3],mych[2]
0x09F2 12	3334	mych[1],mych[0]
0x09F0	09FA	pli
0x09EE	09FE	pi

5. (20 points + 5 points bonus) Design and implement an MSP430 assembly language subroutine *int hex_alpha (int myw)* that processes an integer to determine the number of alphabetical symbols (A to F) in its hexademical representation. For example, hex_alpha(0xABBA)=4, and hex_alpha(0x345A)=1. The main program stores the input myw in the register R12 and the subroutine returns the result in the register R13.

(Bonus 5 points) Design and write a main program that calls the subroutine and displays the result on port 1.

```
#include <msp430.h>                      ; #define controlled include file
            PUBLIC  hex_alpha
            RSEG    CODE                   ; place program in 'CODE' segment
hex_alpha:  PUSH    R4                     ; reserved for the masking result
            PUSH    R6                     ; reserved for the step counter
            CLR     R13                    ; the number of alphabetical symbols
            MOV     #4, R6                 ; step counter is 4 (4 hex digits)
            MOV     R12, R4                ; move myw to R4
gnextdig:   AND     #0x000F, R4            ; get the least significant hex digit
            CMP     #10, R4                ; compare with 10
            JL      lskip                   ; if less than 10 skip
            INC     R13                    ; alphabetical symbol found
lskip:      DEC     R6
            JZ      lend                   ; exit the loop if done
            RRA     R12                    ; shift R12 four times
            RRA     R12
            RRA     R12
            RRA     R12
            MOV     R12, R4                ; move R12 to R4
            JMP     gnextdig               ; go to the next
lend:       POP     R6                     ; restore registers
            POP     R4
            RET
END

/* The main program. */

#include <msp430.h>                    ; define controlled include file

      NAME    main                     ; module name
      PUBLIC  main                     ; make the main label visible
      EXTERN  hex_alpha
                                       ;
      RSEG    CSTACK                   ; pre-declaration of segment
      RSEG    CODE                     ; place program in 'CODE' segment

main: MOV     #SFE(CSTACK), SP         ; set up stack
      MOV.W   #WDTPW+WDTHOLD,&WDTCTL   ; Stop watchdog timer
      BIS.B   #0xFF, &P1DIR            ; set the P1 as output port
      MOV     myw, R12;
      CALL    #hex_alpha
      MOV.B   R13, &P1OUT
      JMP $

myw   DC16   0xAB3D
      END
```