Department of Electrical and Computer Engineering
University of Alabama in Huntsville

CPE 323 – Introduction to Embedded Computer Systems
Midterm Exam

Instructor: Dr. Aleksandar Milenkovic

Date: February 27, 2012

Place: EB 207

Time: 3:55 PM – 5:15 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. Bonus questions are optional. Best wishes.

Question	Points	Score
1	10+3	
2	30	
3	20+5	
4	20	
5	20	
Sum	100+8	

Please print in capitals:

Last name:____________________________

First name: ___________________________

1. (10 points + 3 bonus points) Misc, MSP430

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

1.A. (True | False) (2 points) Assembly language directive "DS32 3" allocates 6 words in memory.

1.B. (True | False) (2 points) Register R0 serves as the program counter.

1.C. (True | False) (2 points) Stack pointer (register R1) always points to the first free location on the top of the stack.

1.D. (True | False) (2 points) The address range of a 1 KB block of data placed in memory at the address 0x0200 is [0x0200 – 0x0800].

1.E. (True | False) (2 points) Instruction ADD R7, R8 requires one 16-bit word to be encoded.

1.F. (bonus, 3 points) How many memory operations (read from memory and write to memory) will be performed during execution of the instruction ADD.W &F000, &F002.

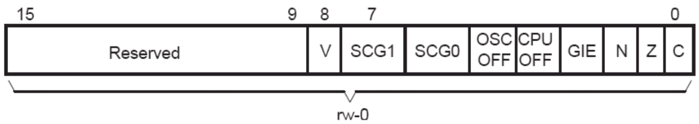
2. (30 points) Assembler (Directives, Instructions, Addressing Modes)

2.A. (10 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), and for character '0' is 48 decimal.

```
        ORG 0xAC00
CBA     DC8 024q, -8, 4, '4', '1'
        EVEN
CBS     DC8 "ABC"
        EVEN
CWA     DC16 18, 0x0230
CLWA    DC32 -5
```

Label	Address [hex]	Memory[15:0] [hex]

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, and the result of the operation. Fill in the empty cells in the table. The initial content of memory is given in the table. Initial value of registers R2, R5, R6, and R7 is as follows: SR=R2=0x0003 (V=0, N=0, Z=1, C=1), R5=0xC001, R6=0xC008. Assume the starting conditions are the same for each question (i.e., always start from initial conditions in memory) and given register values.
Note: Format of the status register (R2) is as follows.



Label	Address [hex]	Memory[15:0] [hex]
	0xC000	0x0504
	0xC002	0xFEEE
TONI	0xC004	0xA821
	0xC006	0x33F4
	0xC008	0xF014
	0xC00A	0x2244
EDE	0xC00C	0xCDDA
	0xC00E	0xEFDD

	Instruction	**Source Addressing Mode**	**Destination Operand Addressing Mode**	**Source Address**	**Dest. Address**	**Result (content of memory location or register)**
(a)	MOV.B &TONI, R5					
(b)	SUBC.B @R6, 5(R5)					
(c)	RRC TONI					
(d)	AND #0x0AC2, -2(R6)					

Notes of setting flags: Instructions that set flags, set N and Z flags as usual. Specific details for C and V are as follows: RRC clears V bit.

3. Analyze assembly program (20 points + 5 bonus points) Consider the following assembly program.

```
1      #include "msp430.h"                      ; #define controlled include file
2              NAME    main                     ; module name
3              PUBLIC  main                     ; make the main label visible
4                                               ; outside this module
5              ORG     0FFFEh
6              DC16    init                     ; set reset vector to 'init' label
7
8              RSEG    CSTACK                   ; pre-declaration of segment
9              RSEG    CODE                     ; place program in 'CODE' segment
       init:   MOV     #SFE(CSTACK), SP         ; set up stack
10     main:   NOP                              ; main program
11             MOV.W   #WDTPW+WDTHOLD,&WDTCTL   ; Stop watchdog timer
12             BIS.B   #0xFF,&P1DIR             ; configure P1.x as output
13             MOV     #greet, R5
14             CLR     R7
15     lnext:  MOV.B   @R5+, R6
16             TST.B   R6
17             JZ      lexit
18             CMP.B   #'A', R6
19             JL      lnext
20             CMP.B   #'Z'+1, R6
21             JGE     lnext
22             INC     R7
23             JMP     lnext
24     lexit:  MOV.B   R7, &P1OUT
25             JMP     $
26     greet:  DC8     "HELLO Midterm!";
27     end:
28             END
```

3.A. (2 points) How many bytes is used to store the string at label greet?

3.B. (3 points) What does the instruction in line 13 do?

3.C. (10 points) What does this program do? Add code comments (lines 13-24).

3.D. (5 points) What is the value on P1OUT at the end of the program?

3.E. (bonus, 5 points) Estimate execution time of the code segment until statement in line 25 is reached. Assume the following: on average each instruction executed takes 2 clock cycles and the clock frequency is 1 MHz. Show your work. ascii(space)=0x20, ascii(‘!’)=0x21, ascii(‘A’)=0x41.

4. Design assembly program (20 points) Design and write an MSP430 assembly language subroutine *unsigned int max(unsigned int *a, unsigned int n)* that returns the maximum of an array of *n* unsigned integers.
What does the main program do with the maximum? How do we pass the input parameters (array starting address and array length) to the subroutine? How does the subroutine return the maximum?

```
#include "msp430.h"                     ; #define controlled include file

        NAME    main                    ; module name

        PUBLIC  main                    ; make the main label visible
                                        ; outside this module
        ORG     0FFFEh
        DC16    init                    ; set reset vector to 'init' label

        RSEG    CSTACK                  ; pre-declaration of segment
        RSEG    CODE                    ; place program in 'CODE' segment

init:   MOV     #SFE(CSTACK), SP        ; set up stack

main:   NOP                             ; main program
        MOV.W   #WDTPW+WDTHOLD,&WDTCTL  ; Stop watchdog timer
        BIS.B   #0xFF, P1DIR            ; P1 is configured as output
        BIS.B   #0xFF, P2DIR            ; P2 is configure as output
        MOV.W   #myarr, R5              ; R5 has the address of myarr
        MOV     #myn, R6                ; R6 has the address of myn
        SUB     R5, R6                  ;
        RRA     R6                      ;
        PUSH    R6                      ;
        SUB     #2, SP                  ;
        CALL    #maxel                  ; call subroutine
        MOV.B   @SP, P1OUT              ;
        MOV.B   1(SP), P2OUT            ;
        ADD     #4, SP                  ; free stack

        JMP     $

myarr: DC16  7, 12, 45, 32, 27, 22, 112, 63000, 22
myn:

maxel:
```

5. (20 points, C language) Consider the following C program. Assume that the register SP at the beginning points to 0x1000. Answer the following questions. Assume all variables are allocated on the stack, and in the order as they appear in the program.

5.A. (10 points) Illustrate the content of the stack at the moment before the statement at line 8 is executed. ascii('1') = 0x31.

5.B. (10 points) Comment the code (lines 8 – 13) indicating the result of each statement. Illustrate the content of the stack at the end of execution of the statement in line 13.

```
1   int main( void )  {
2     volatile unsigned int a[3] = {3,4,5};
3     volatile int b = -4;
4     volatile long int c = -5;
5     volatile char d[2] = {'1','2'};
6     volatile unsigned int *p;
7
8     p = a;
9     p = p - 2;
10    *p = *p + 4;
11    p++;
12    *p = 11;
13    a[0] = *p + a[1];
    }
```

A.

Address	M[15..0]	Comment
0x1000		OTOS

B.

Address	M[15..0]	Comment
0x1000		OTOS