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 13, 2014

Place: EB 207

Time: 3:55 PM – 05:15 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. 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 4" allocates 8 16-bit words in memory.

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

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

1.E. (True | False) (2 points) Stack grows from lower to higher addresses in memory.

1.F. (2 points) How many memory operations (reads from memory and writes to memory) occurs during execution of the instruction MOV.B #45, &P1OUT?

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 2 KB block of data placed in memory at the address 0x0A00? Fill in the blanks.

[0x0A00__ – 0x11FF]

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

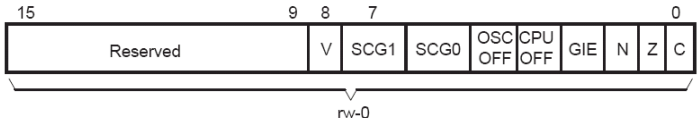
2.A. (5 points) Show the word-wide **HEXADECIMAL** content of memory corresponding to the following sequence of assembler directives. ASCII code for character 'A' is 65 (decimal), and for character '0' is 48 decimal.

```
// suffix q stands for octal, suffix b stands for binary, prefix 0x for hex
          ORG 0xC000
C1        DC8 021q, 0xA0, 00000011b
          EVEN
C2        DC8 'A', "BC"
          EVEN
C3        DC32 -2
```

Label	Address [hex]	Memory[15:0] [hex]
C1	0xC000	0xA011
	0xC002	0x??03
C2	0xC004	0x4241
	0xC006	0x0043
C3	0xC008	0xFFFE
	0xC00A	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=0x0000 (V=0, N=0, Z=0, C=0), 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	0xA862
	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; Flags (C,V,Z, and N) ARE NOT required.**
(a)	DADD.B @R6, &TONI	2	Register Indirect	Absolute	C006	C004	byte operation: 0x62 + 0x44 = 0x06 => write the byte to M[C004] = 0x06
(b)	SUBC DEN, R6	2	Symbolic	Register direct	0xC00A	-	word operation (dst + not. src + C) src = M[0xC00A] = 0x2244 dst = R6 = 0xC006 => result: C006+DDBB+0 = 0x9D11, R6 = 0x9DC1 Flags: R2=0x0003, C=1, Z=0, N=1, V=0
(c)	XOR.W @R6+, 3(R5)	2	Register indirect with autoincrement	Indexed	C006	C006	Src .xor. Dst src = 0x3344 dst = 0x3344 M[C006] = 0x0000 R6 = 0xC008 C = 0, V = 0, N = 0, Z = 1

3. Analyze assembly program (20 points)

Consider the following code segment.

```
01        CLR     R12
02        MOV.B   myb, R6                 ;
03        MOV     #0x0001, R5             ;
04        MOV     #8, R7                  ;
05 lnxt:  BIT.B   R5, R6                  ;
06        JNZ     lskip                   ;
07        INC     R12                     ;
08 lskip:RRA.B    R6                      ;
09        DEC     R7                      ;
10        JNZ     lnxt                    ;
11        MOV.B   R12, P1OUT
12
13 myb:   dc8  1000_1110b     ; suffix b denotes binary value
```

3.A. (2 points) How many bytes is allocated by the assembly directive in line 13?

1 byte (0x8E).

3.B. (2 points) What is the content of register R6 after the instruction in line 02 is completed?

Register R6=0x008E (the content of the word at label myb).

3.C. (10 points) What does this code segment do? Explain your answer.

This code segment counts the number of bits set to a '0' in the constant myb. The result (4) is displayed on P1.

3.D. (2 points) What is the value of P1OUT at the end of the program.

P1OUT = 0x04.

3.E. (4 points) Calculate the total execution time in seconds for the code sequence from above (line 01 – line 11). We know the following: the average CPI is 2 clocks per instruction. Assume the clock frequency is 2 MHz. What is MIPS rate for this code?

*IC = 4 + 4*6 +4*5 + 1= 49 instructions*
*ExTime = IC*CPI/ClockFreq = 49*2/2*10^6= 49 us*
MIPS = 2/2 = 1 MIPS

4. (20 points, C language) Consider the following C program. Assume that the register SP at the beginning points to 0x0800. 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.

```
1   int main( void )  {
2     volatile unsigned long int a = 4;
3     volatile unsigned int c = -1, d = 2;
4     volatile char mych = {'0', '1', '2', '4'};
5     volatile unsigned long int *lpa = &a;
6
7     lpa = lpa - 2;    // lpa = lpa - 2*sizeof(long int)
8     *lpa = *lpa + 4;    //
11  }
```

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.	4 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 2-5.	14 bytes
4	Value of mych[0] after initialization performed in line 4.	'0'/ 0x30
5	Address of variable a (&a).	0x07FC
7	Value of lpa at the moment after the statement in line 5 is executed.	0x07FC
8	Value of lpa at the moment after the statement in line 7 is executed.	0x07F4
9	Value of *lpa at the moment after the statement in line 8 is executed.	0x33323134
10	Value of mych[0] at the moment after the statement in line 8 is executed.	'4'

0x0800	TOS	Comments
0x07FE	0000	a.upper
0x07FC	0004	a.lower
0x07FA	FFFF	C
0x07F8	0002	D
0x07F6	3432	mych[3],mych[2]
0x07F4	3130	mych[1],mych[0]
0x07F2	07F4	lpa

5. (20 points) Design and write an MSP430 assembly language subroutine that returns the number of lower case letters (‘a’ – ‘z’) in an ASCII string, i.e., `unsigned int num_lcaseletter (char *mystring)`. The subroutine expects the starting address of the input string in the register R12 and returns the result in the register R13.

What value is returned by the subroutine for the following input string: `char *mystring = "An example string."`?

(Bonus 5 points) Design and write a main program that calls the subroutine and displays the number of lower case letters on ports P1 and P2.

```
        PUBLIC num_lcaseletter
        RSEG CODE
num_lcaseletter: ; write your code here
        ; save the registers on the stack
        PUSH    R7                      ; to keep current element
        CLR     R13                     ; set the output to zero
lnext:  MOV.B   @R12+, R7               ; read a(i)
        CMP.B   #0, R7                  ; is it a NULL character
        JZ     lend                     ; if yes, go to the end
        CMP.B   #'a', R7                 ;
        JL      lnext
        CMP.B   #'z'+1, R7
        JG      lnext
        INC     R13
        JMP     lnext
lend:   POP R7
        RET
        END
```

An example main program:

```
#include "msp430.h"                     ; #define controlled include file
        NAME    main                    ; module name
        PUBLIC  main                    ; make the main label visible
                                        ; outside this module
        EXTERN  num_lcaseletter
        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:   MOV.W   #WDTPW+WDTHOLD,&WDTCTL  ; Stop watchdog timer
        BIS.B   #0xFF,&P1DIR            ; configure P1.x as output
        BIS.B   #0xFF,&P2DIR            ; configure P2.x as output
        MOV.W   #mystring, R12          ; R12 gets the address of the character string mystring
        CALL    #num_lcaseletter
        MOV.B   R13, P1OUT              ; display lower byte on P1OUT
        SWPB    R12
        MOV.B   R13, P2OUT              ; display higher byte on P2OUT
        JMP     $

mystring:    DC8     "An example string." ; a test string
        END
```