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

IAR Define storage .bss 12, 2

.usect .bss align

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.

6 16 bits bit 3 of em

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.

Placement, SP last Full location on tos

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].

look at range 1kB= 2^{10} bytes

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.

1w
1w F000
1w F002

3 reads - instruction
2 reads - op-fetch
1 write -

M[F002] ← M[F002] + M[F000]

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.

origin moving assembler to address

R3 Const gen

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

Cstring ABC

DC8 -> Define Constant 8-bit long

0249 0001 0,100

Label	Address [hex]	Memory[15:0] [hex]	
CBA	AC00	-8 F8	14 0249
	AC02	34	4
	04	?	31
CBS	06	42	41
	08	00	43
CWA	0A	00	12
	0C	02	30
ClWA	0E	FF	FB
	10	FF	FF

mov.w #CBS, R4

R4= AC06

mov.w CBS, R5

R5 = 4241

R5← M[CBS] ->AC07!

mov.B CBS+1, R6

R6= 0x0042

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.

15–9	8	7							0
Reserved	V	SCG1	SCG0	OSC OFF	CPU OFF	GIE	N	Z	C

rw-0

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

F4 14

	Instruction	Source Addressing Mode	Destination Operand Addressing Mode	Source Address	Dest. Address	Result (content of memory location or register)
(a)	MOV.B &TONI, R5	Abs	Register	0xC004	0xC001	R5 = 0x0021 R2 = 0x0003
(b)	SUBC.B @R6, 5(R5) just change byte @ C006	Indirect Register	Indexed	C008	C006	dst + .src + C F4 + EB + 1 = E0 M[C006] = 0x33E0 C = 1, N = 1 Z = 0, V = 0
(c)	RRC TONI	/	register	/	0xC004	M[C004] ← D410 C = 1, Z = 0 N = 1, V = ?
(d)	AND #0x0AC2, -2(R6)	Immediate	Indexed	?	C006	33F4 0AC2 02C0 M[C006] ← 02C0

A821

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.

1

1010 1000 0010 0001 1

1101 0100 0001 0000
D 4 1 0

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       6     BIS.B    #0xFF,&P1DIR            ; configure P1.x as output
13             MOV      #greet, R5
14             CLR      R7   R7=0
15     lnext:  MOV.B    @R5+, R6   R6 gets next char, H first R5 now e.
16             TST.B    R6
17             JZ       lexit
18             CMP.B    #'A', R6   A vs. R6
19             JL       lnext      less than A?
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!";   .cstring "Hello Midterm!"
27     end:
28             END      R5
```

num of upp case letters in string

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

14 char + Null = 15 bytes.

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

Address of string at label greet in R5

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

# of upper case letter in string in greet and displays # on P1out

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

P1OUT = 6

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.

CPI = 2cc

IC = 6 + [$\underset{\text{upper}}{6 \times 9}$ + $\underset{\text{special}}{2 \times 5}$ + $\underset{\text{lower}}{6 \times 7}$] + 1 × 3 = 116

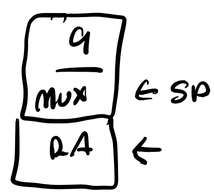
ET = IC × CPI × CCT

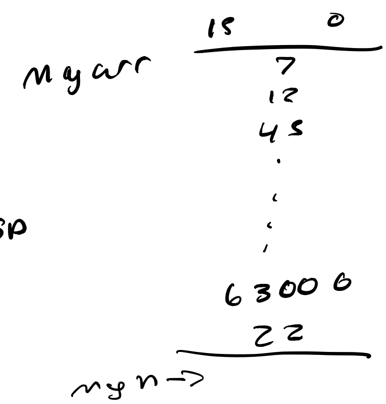
= 116 × 2 × 1 μs

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:
```

→9 pushed on stack





R5 contains address of my arr

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;` address of A. 09FA
9	`p = p - 2;` p points to upper of C. 09FC
10	`*p = *p + 4;` FFFF +4 = 0010, (upper = 0003
11	`p++;` 09F6 → 09F8 (b)
12	`*p = 11;` b= 000b = 11
13	`a[0] = *p + a[1];`
	`}` 11+ 4=15= 000F

A.

Address	M[15..0]	Comment
0x1000		OTOS
0x09FE	0005	a [2]
09FC	0004	a [1]
09FA	0003	a [0]
09F8	FFFC	b
09F6	FFFF	c
09F4	FFFA	c
09F2	3231	d
09F0	?	

B.

Address	M[15..0]	Comment
0x1000		OTOS
0x09FE	0005	a [2]
09FC	0004	a [1]
09FA	0003	a [0]
09F8	FFFC	b
09F6	~~FFFF~~ 0003	c
09F4	FFFA	c
09F2	3231	d
09F0	09EA	

B.

Address	M[15..0]	Comment
0x1000		OTOS
0x09FE	0005	a [2]
09FC	0004	a [1]
09FA	0003	a [0]
09F8	FFFC	b
09F6	FFFF	c
09F4	FFFA	c
09F2	3231	d
09F0	?	