16.317: Microprocessor Systems Design I

Fall 2014

Exam 2
November 5, 2014

Name: _____ ID #: _____

For this exam, you may use a calculator and one 8.5" x 11" double-sided page of notes. All other electronic devices (e.g., cellular phones, laptops, tablets) are prohibited. If you have a cellular phone, please turn it off prior to the start of the exam to avoid distracting other students.

The exam contains 4 questions for a total of 100 points. Please answer the questions in the spaces provided. If you need additional space, use the back of the page on which the question is written and clearly indicate that you have done so.

Please note that Question 4 has three parts, but you are only required to complete two of the three parts. You may complete all three parts for up to 10 points of extra credit. If you do so, **please clearly indicate which part is the extra one—I will assume it is part (c) if you mark none of them.**

Note also that your solutions to Question 4 will be short sequences of code, not subroutines. **You do not have to write any code to deal with the stack when solving Question 4.**

You will be provided with six pages (3 double-sided sheets) of reference material for the exam: a list of the x86 instructions and condition codes we have covered thus far, a description of subroutine calling conventions, and a list of the PIC 16F1829 instructions we have covered thus far. You do not have to submit these pages when you turn in your exam.

You will have 50 minutes to complete this exam.

Q1: Multiple choice	/ 16
Q2: Reading PIC assembly	/ 16
Q3: Subroutines; HLL → assembly	/ 28
Q4: Conditional instructions	/ 40
TOTAL SCORE	/ 100
EXTRA CREDIT	/ 10

1. (16 points, 4 points per part) ***Multiple choice***
For each of the multiple choice questions below, clearly indicate your response by circling or underlining the single choice you think best answers the question.

a. Which of the following statements about interrupts and exceptions are true?

   A. All external interrupts can be disabled so that an interrupt service routine is not called when an external device asserts an interrupt input pin.

   B. An interrupt vector is the starting address of an interrupt service routine. Interrupt vectors are typically stored in a table located in memory.

   C. When an interrupt occurs, the interrupt service routine is responsible for saving the processor state (all registers, including the flags).

   D. Most processors have both dedicated interrupt vectors for specific types of interrupts and general vectors to be used with user-defined interrupts.

   i. Only A

   ii. Only B

   iii. A and C

   iv. B and D

   v. All of the above (A, B, C, and D)

b. If the working register is set to `0x10` and a file register, `x`, is set to `0x08`, what is the result of the instruction `subwf x, W`?

   i. `W = 0x08`

   ii. `x = 0x08`

   iii. `W = 0xF8`

   iv. `x = 0xF8`

1 (continued)

c. Which of the following instructions will set the zero bit (Z) to 1 if the file register `x` is equal to 0?

A. `movwf  x`

B. `movf  x, F`

C. `bcf  x, 0`

D. `addlw  0x00`

i. Only A

ii. Only B

iii. A and B

iv. A, B, and C

v. A, B, C, and D

d. Which of the following instructions can <u>always</u> be used to set (in other words, change to 1) the upper four bits of the working register, W, while leaving the lower four bits of the register unchanged?

i. `clrw`

ii. `sublw  0x0F`

iii. `iorlw  0xF0`

iv. `xorlw  0x0F`

v. `andlw  0xF0`

2. (16 points) ***Reading PIC assembly***

Show the result of each PIC 16F1829 instruction in the sequences below. Be sure to show the state of the carry (C) bit for any shift or rotate operations.

a.
```
cblock 0x75
   x
endc
movlw   0x79


movwf   x


comf    x, W


lslf    x, F


addwf   x, W


btfss   x, 6


sublw   0x99


xorwf   x, F
```

3. (28 points) ***Subroutines; HLL → assembly***

The following questions deal with the register and memory contents shown below. Note that:

- These values represent the state of the registers and stack immediately after the stack frame has been set up for the current functions.
- The values shown in memory make up the entire stack frame for the current function.

EAX:	0x0000ABBA
EBX:	0x00001234
ECX:	0x00005099
EDX:	0xFFFFFFFF
ESI:	0x11340550
EDI:	0x11340590
ESP:	0x11320140
EBP:	0x1132014C

Address	
0x11320140	0x00000005
0x11320144	0x0000000A
0x11320148	0xFFFF0000
0x1132014C	0x11320164
0x11320150	0x20010550
0x11320154	0x00000002
0x11320158	0x08675309
0x1132015C	0x00000088
0x11320160	0x00197800

a. (3 points) What is the return address for this function?

b. (5 points) How many arguments does this function take, and what are their values? Indicate which of the arguments is passed to the function first (for example, when calling a function `f(13, 14, 15)`, the value 13 is passed first). Assume each argument uses four bytes.

3 (continued)

c. (5 points) If we assume that each local variable uses four bytes, how many local variables are declared in this function? Explain your answer.

d. (15 points) A partially completed x86 function is written below. Complete the function by writing the appropriate instructions in the blank spaces provided. The comments next to each blank or instruction describe the purpose of that instruction. Assume that the function takes two arguments (`v1` and `v2`, in that order) and contains a single local integer variable, `x`.

```
f  PROC                          ; Start of function f
   push    ebp                   ; Save ebp
   mov     ebp, esp              ; Copy ebp to esp

   sub     esp, 4                ; Create space on stack for x

   mov     ebx, DWORD PTR 8[ebp] ; ebx = v1


   _____________________________ ; ebx = v1 - v2


   _____________________________ ; x = ebx = v1 - v2 (copy ebx
                                 ;    to memory location for x)
   sll     ebx, 4                ; ebx = ebx << 4 = x << 4


   _____________________________ ; ebx = ebx – x = (x << 4) – x


   _____________________________ ; Clear space allocated for
                                 ;     local variable
   pop     ebp                   ; Restore ebp

   _____________________________ ; Return from subroutine
f  ENDP
```

4. (40 points) ***Conditional instructions***

For each part of this problem, write a short x86 code sequence that performs the specified operation. **CHOOSE ANY TWO OF THE THREE PARTS** and fill in the space provided with appropriate code. **You may complete all three parts for up to 10 points of extra credit, but must clearly indicate which part is the extra one—I will assume it is part (c) if you mark none of them.**

Note also that your solutions to this question will be short sequences of code, not subroutines. **You do not have to write any code to deal with the stack when solving these problems.**

a. Implement the following conditional statement. You may assume that "X", "Y", and "Z" refer to 16-bit variables stored in memory, which can be directly accessed using those names (for example, `MOV AX, X` would move the contents of variable "X" to the register AX).

```
if ((X == AX) && (Y == BX) {
   Z = Z / 2;
}
else {
   Z = Z * 4;
   if (Z > X)
      X = Z;
}
```

4 (continued)

b. Implement the following loop. Assume that `ARR` is an array of forty 32-bit values. The starting address of this array is in the register `SI` when the loop starts—you can use that register to help you access values within the array.

```
for (i = 40; i > 0; i = i - 2) {
   ARR[i-1] = ARR[i-2] + i;
   ARR[i-2] = ARR[i-1] – i;
}
```

4 (continued)

c. Implement the following loop. As in part (a), assume "X", "Y", and "Z" are 16-bit variables in memory that can be accessed by name. Recall that a while loop is a more general type of loop than the for loop seen in part (b)—a while loop simply repeats the loop body as long as the condition tested at the beginning of the loop is true.

```
while (X >= Y) {
   Y = Y + Z – 1;
   X = X – Z + 1;
}
```