UNCA CSCI 255
Final Exam Fall 2013
9 December, 2013

This is a closed book and closed notes exam. It is to be turned in by 10:30 AM. Calculators, PDA's, cell phones, and any other electronic or communication devices may not be used during this exam.
If you want partial credit for imperfect answers, explain the reason for your answer!

Problem 1 (6 points) Decimal to two's complement conversion
Convert the following six signed decimal numbers into seven-bit two's complement representation. Some of these numbers may be outside the range of representation for seven-bit two's complement numbers. Write "out-of-range" for those cases.

0	-1
-100	64
42	-42

Problem 2 (4 points) Two's complement to decimal conversion
Convert the following four seven-bit two's complement numbers into signed decimal representation.

0110110	1001001
1111110	1111111

Problem 3 (6 points) Adding
Add the following pairs of seven-bit two's complement numbers **and indicate which additions result in an overflow by writing one of "overflow" or "no overflow" in each box**. You must write either "overflow" or "no overflow" in each box in addition to the result of the addition.

0110101 + 0000011	0110100 + 0011100
0110011 + 1001101	0010000 + 1110000
1010111 + 1010001	1111111 + 1111111

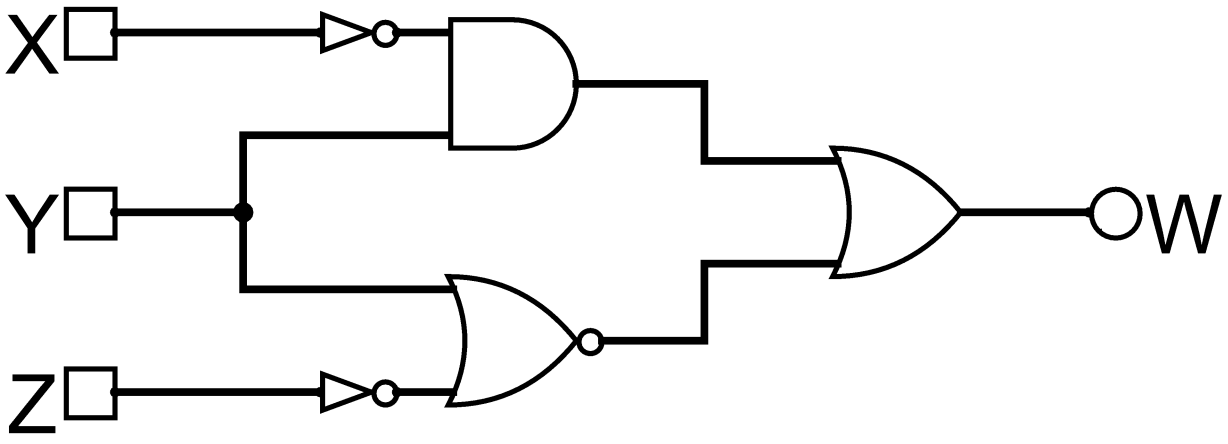
Problem 4 (4 points) Memories
An 128 MB memory has a 32-bit word size. How many words are contained in this memory?

A memory has 4 M words. Each word contains 8 bits. How many bytes are contained in this memory?

How many bits are required to address the 4 M words of this memory?

Problem 5 (7 points) Digital logic to truth table
A gate-level circuit is shown below with three inputs on the left and a single output on the right. Complete the truth table so that it corresponds to this digital logic circuit.

X	Y	Z	W
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	



Problem 6 (3 points) Digital logic to Boolean expression
Write a Boolean expression that corresponds to the logic circuit shown in Problem 5. You can build on your Problem 5 answer if that seems appropriate.

Problem 7 (7 points) Truth table to digital logic
Draw a logic circuit at the gate level that will implement the following truth table, where X, Y, and Z are inputs and W is the single output.

X	Y	Z	W
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Problem 8 (3 points) Truth table to Boolean expression
Write a Boolean expression that corresponds to the truth table shown in Problem 7. You can build on your Problem 7 answer if that seems appropriate.

Problem 9 (4 points) Boolean expression to truth table
Complete the truth table on the left below so that it corresponds to the following Boolean equation

$$W = X Y + \overline{(Y + Z)}$$

If you prefer that your inversions be primes, you can think of the equation as

$$W = X Y + (Y + Z)'$$

Or, if you really like Java conditional expressions, you can go with

```
W = X && Y || !(Y || Z)
```

Problem 10 (6 points) Boolean expression to digital logic
On the remainder of this page, draw a logic circuit at the gate level that will implement the Boolean equation given in Problem 9. You can build on your Problem 9 answer if that seems appropriate.

X	Y	Z	W
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Problem 11 (3 points) Boolean expression to digital logic
A finite state machine (FSM) is described by the following state table.

present state	input	next state
α	0	β
α	1	α
β	0	γ
β	1	α
γ	0	δ
γ	1	α
δ	0	δ
δ	1	δ

Assuming that the finite state machine is started in state α and the finite state machine is given the input sequence **0 0 1 0 1 1 0 0 0 1 0**, what state is the finite state machine in after each input?

Just fill in the blanks with the state that will be entered after the input immediately to the left of the blank arrives. Keep in mind the state machine is not reset after each input. It makes a transition from the preceding state.

You do need to write any outputs. They aren't even given in the problem description.

α 0 _ 0 _ 1 _ 0 _ 1 _ 1 _ 0 _ 0 _ 0 _ 1 _ 0 _

Just to be clear on this. The first blank on the left above should be filled in with the new state that the finite state machine enters after it receives a 0 when the present state is α.

Problem 12 (6 points)
In the tables below the values of several PIC working registers and memory locations (file registers) are given.

Registers	
W0	0x0906
W1	0xBEEF
W2	0x0828
W3	0x0902
W4	0x08F0

Memory	
0x0900	0x00FF
0x0902	0x0000
0x0904	0x0200
0x0906	0x0800
0x0908	0x0A00

For the following six PIC instructions, list any registers or memory locations that are modified by the instruction along with the value stored into the register or location. Assume that each instruction is executed using the values shown above. That is, the instruction executions are *not* sequential. You can give the new value in either decimal or hexadecimal.

`ADD W0,#-8,W2`
`AND #0x77,W2`
`IOR #0x77,W1`
`MOV #0xFADE,W3`
`MOV W3,[--W0]`
`MOV [W0++],[++W3]`

Problem 13 (6 points)
In the left column there are six PIC24 instructions. In the right column write the 24 bits needed to encode each instruction in the PIC24 instruction set.

`add W2,W3,W5`	_ _
`add #15,W7`	_ _
`add PCL,WREG`	_ _
`mov W5,PCL`	_ _
`mov W10,[W2]`	_ _
`mov #-5,W3`	_ _

Setup for problems on next two pages
In the problems on the next two pages, you are going to write sections of PIC24 assembly code corresponding to C statements. In writing this code, assume that `i` is a 16-bit unsigned integer, that `sum` is a 16-bit *signed* integer and that V is an array of 100 16-bit *signed* integers. These could be declared in C as follows:

```
uint16_t  i ;
int16_t   sum ;
int16_t   V[100] ;
```

In PIC24 assembly code, space for these would be allocated using the `.space` directive is follows:

```
i:      .space  2
sum:    .space  2
V:      .space  200
```

In each of the problems, a section of C code is given. You are to translate that code into PIC24 code. You are free to use `WREG` and any of the sixteen PIC24 working registers in your answers. In some instances you will also need to use and define labels.

Look at the PIC24 instruction guide carefully to make sure you don't make any beginner errors in the excitement of working these problems. Often you will need to load C variables into a register before using them in a calculation.

Problem 14 (6 points)

```
if (i > 5) {
   i++ ;
}
if (sum > 5) {
   sum++ ;
}
```

Problem 15 (10 points)

```
sum = 0 ;
for (i=0; i<100; ++i) {
   if (V[i] > 0) {
      sum = sum + V[i] ;
   }
}
```

Problem 16 (6 points)

In the last few labs of the semester, we wrote PIC C code to perform GPIO (general-purpose input/output) with the PIC processor. In this problem, you are going to write a bit more PIC C. Remember the following:

- `_LATB6` is used to set the value written to port `B6`
- `__delay_ms` will delay for milliseconds
- `__delay_us` will delay for microseconds

Assume that port `B6` has already been set up for digital output and that an LED and resistor have been properly attached to the port with the cathode of the LED grounded.

Write a loop that repeats the following sequence *forever*, once every three seconds.

- The LED is completely off for one second.
- The LED is on a bit brighter, about one-half power, for one second
- The LED is on at full power for one second

Problem 17 (3 points)
In a recent lab, you were given the following `struct` definition with an associated `typedef`.

```
struct noteInfo {
  uint16_t   frequency ; /* frequency in Hz */
  uint16_t   duration ;  /* duration in mSec */
} ;
typedef struct noteInfo Note ;
```

In the space below, declare a variable called `middleCNote` of type `Note` and initialize `Note` for a frequency of 262 Hertz and a duration of one second.

In the lab we also used a C function **playTone** with the following prototype:

```
void playTone(const Note *tone) ;
```

Write a line of C code to pass your `middleCNote` variable to `playTone`.

Finally, what C expressions would be used by C programmer to refer to the frequency and duration of the `tone` and `duration` parameter inside the `playTone` function?

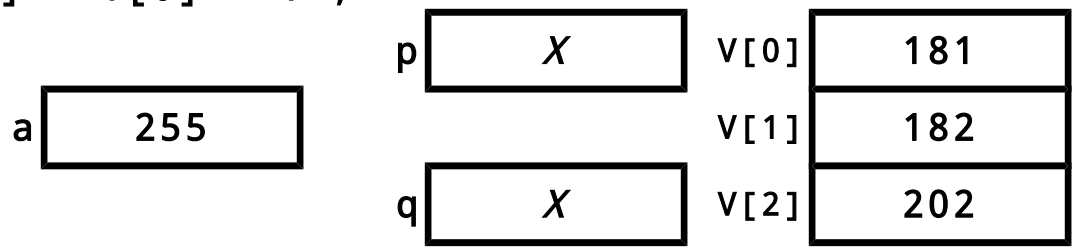
Problem 18 (10 points)
In this question, you are to fill in boxes representing the following C integer or pointer variables to show their values after each of seven sections of C code are executed. You should consider all the sections as being independently executed after the following declaration and initialization statements:

```
int    a = 255 ;
int    V[3] = {181, 182, 202} ;
int    *p = NULL ;
int    *q = NULL ;
```

As you might guess, `null` in Java is similar to `NULL` in C. (The `NULL` of C is derived from the `nil` of LISP.) Draw the value `NULL` with a little X. Don't just leave the pointer variable boxes empty.

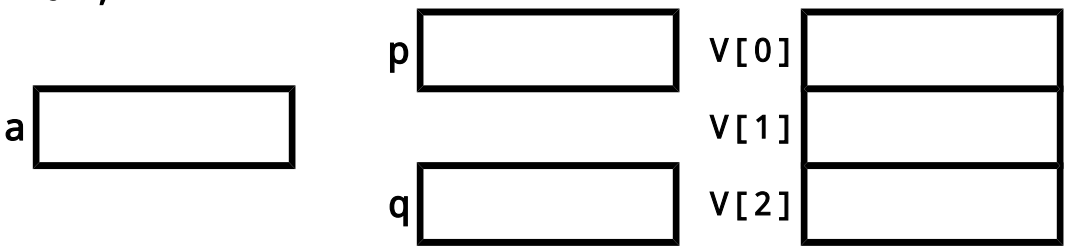
Code section @ (the starting point)

```
V[1] = V[0] + 1 ;
```



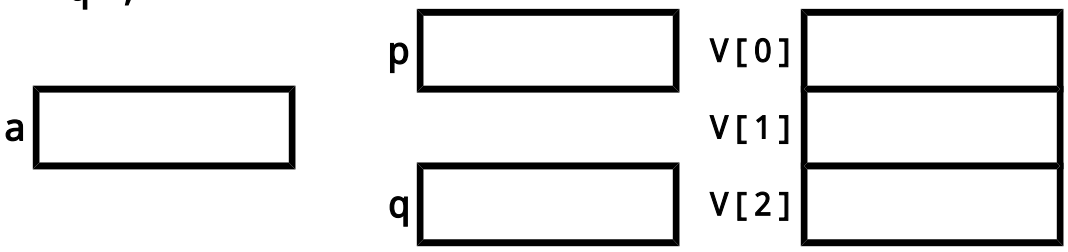
Code section A

```
p = V ;
q = &V[2] ;
*p = 5 ;
*q = 6 ;
```



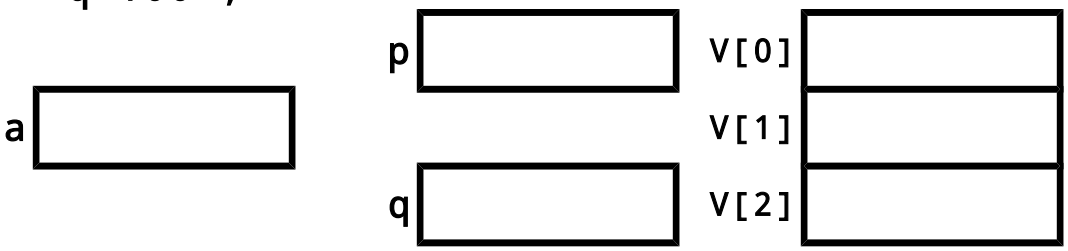
Code section B

```
p = &a ;
q = &V[1] ;
*p = *q ;
```



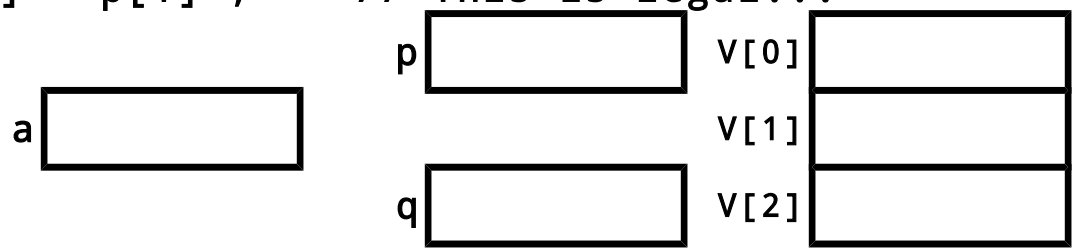
Code section C

```
p = &V[0] ;
q = p ;
*p = *q+100 ;
```



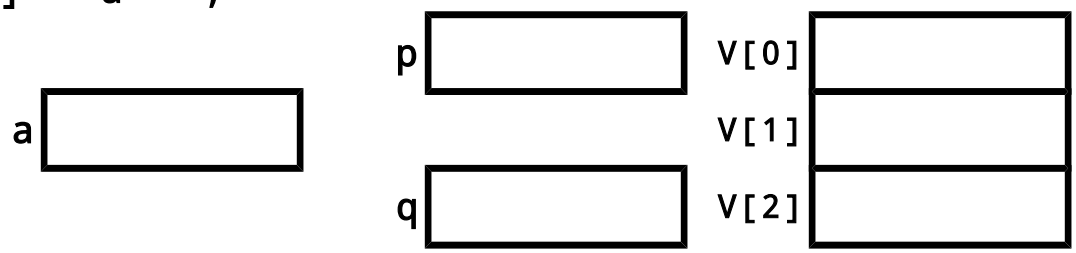
Code section D

```
p = &V[0] ;
q = &V[1] ;
q[1] = p[1] ;    // This is legal...
```



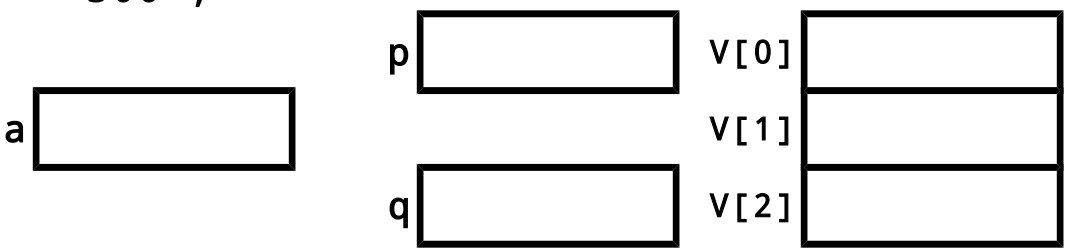
Code section E

```
V[0] = a++ ;
V[1] = ++a ;
V[2] = a++ ;
```



Code section F

```
p = &V[0] ;
q = &V[1] ;
++p* = 200 ;
q++* = 300 ;
```



Code section G

```
p = &V[0] ;
q = &V[1] ;
a = q-p ;
V[2] = *p-*q ;
```