UNCA CSCI 255
Final Exam Fall 2014
8 December, 2014

This is a closed book and closed notes exam. It is to be turned in by 2:00 PM.

Communication with anyone other than the instructor is not allowed during the exam. Furthermore, calculators, cell phones, and any other electronic or communication devices may not be used during this exam. Anyone needing a break during exams must leave their exam with the instructor. Cell phones or computers may not be used during breaks.

If you want partial credit for imperfect answers, explain the reason for your answer!

Name:_____

Problem 1 (4 points) Decimal to two's complement conversion
Convert the following four signed decimal numbers into six-bit *two's complement* representation. Some of these numbers may be outside the range of representation for six-bit two's complement numbers. Write "out-of-range" for those cases.

-32	**-1**
20	**40**

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

010010	**011110**
101011	**111111**

Problem 3 (2 points) Binary arithmetic
Perform the following operations and express the result as it should be for CSCI 255. (Remember Problem 2 of homework 4. Keep it simple!)

8k * 256
32M / 128

Problem 4 (4 points) Adding signed numbers
Add the following pairs of six-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.

010000 **+ 110000**	**011000** **+ 011000**
111000 **+ 111000**	**101000** **+ 101000**

Problem 5 (2 points) Memories

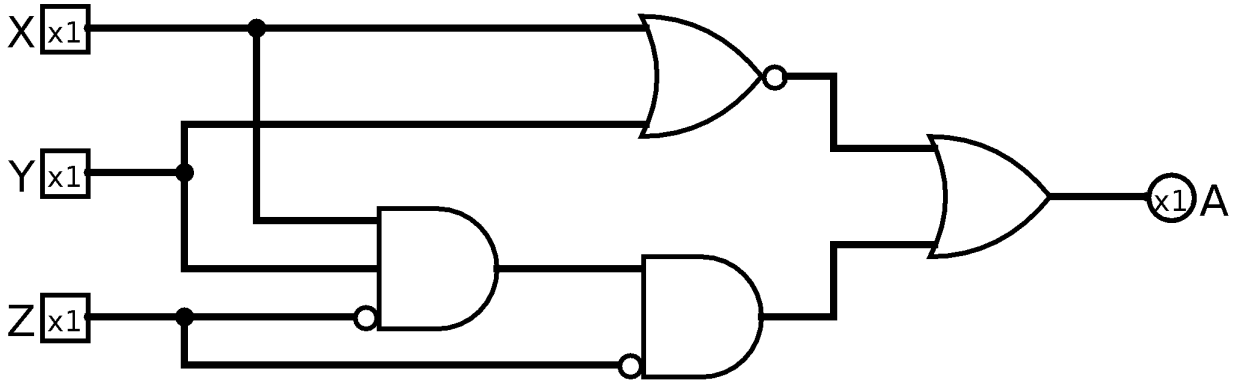
Consider a memory with 2 G bytes (16 G bits) where each word is 32 bits.

How many words are contained in this memory?

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

Remember to express your answers in the CSCI 255 way.

Problem 6 (6 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	A
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 7 (2 points) Digital logic to Boolean expression
Write a Boolean expression that corresponds to the logic circuit shown in Problem 6. You can build on your Problem 8 answer if that seems appropriate.

Problem 8 (6 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 A is the single output.

X	Y	Z	A
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

Problem 9 (2 points) Truth table to Boolean expression
Write a Boolean expression that corresponds to the truth table shown in Problem 8. You can build on your Problem 8 answer if that seems appropriate.

Problem 10 (4 points) Boolean expression to truth table
Complete the truth table on the left below so that it corresponds to the following Boolean equation

$$A = X \overline{Y + Z} + \overline{X}\, Y\, \overline{Z} + Y\, Z$$

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

$$A = X\,(Y + Z)' + X'\,Y\,Z' + Y\,Z$$

Or, if you really like Java and C expressions, you can go with

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

Problem 11 (4 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 10. You can build on your Problem 10 answer if that seems appropriate.

X	Y	Z	A
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 12 (10 points)
Below is a section of seven MIPS32 instructions written in MIP32 assembly language.

```
            lw      $t3,16($sp)
            add     $t5,$zero,$zero
Again:      andi    $t6,$t3,1
            add     $t5,$t5,$t6
            srl     $t3,$t3,1
            bne     $t3,$zero,Again
            nop
```

In the table below these seven MIPS32 instructions are repeated. Write the 32 bits needed to encode each of the non-nop instructions in the MIPS32 instruction set.

`lw $t3,16($sp)`   _
`add $t5,$zero,$zero`   _
`Again: andi $t6,$t3,1`   _
`add $t5,$t5,$t6`   _
`srl $t3,$t3,1`   _
`bne $t3,$zero,Again`   _
`nop`   00000000000000000000000000000000

Problem 13 (8 points)
Let's have a few short answer questions. The answer with the lowest grade will be dropped when computing the earned points for this problem. (That is, you can skip one.)

13A How do you set a breakpoint in the MPLAB X IDE?

13B How do you can examine the value of `$t5` when debugging in the MPLAB X IDE?

13C How do you use the PIC32 PORTB *special function register* to read the input value on port B3? (Writing the C statement would be nice.)

13D How were pullups used in the class?

13E What does it mean to “program a PIC32 with a PICkit 3”?

13F What are a couple of ways to reduce power usage in a microcontroller?

Problem 14 (2 points)
In the last lab, you were given the following `struct` definition with an associated `typedef`.

```
struct noteInfo {
   int  frequency ;           /* frequency in Hz */
   long duration ;            /* duration in mSec */
} ;
```

Complete the following function, called `chipmunk`, that doubles the `frequency` and halves the `duration` of a `struct noteInfo` passed using a pointer.

```
void chipmunk(struct noteInfo *note) {



}
```

Problem 15 (8 points)
In the left column, there are some tricky C expressions. Write their values in the right column. If the values are integers, express them in base 10.

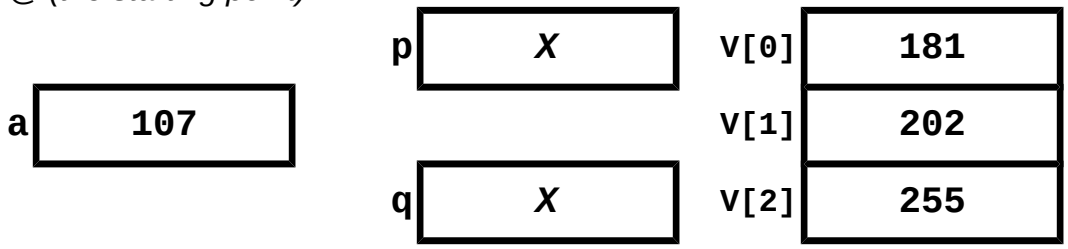
17 % 10	
17 / 10	
10 / 5 * 2	
(17, 15) + 5	
17 & 14	
17 | 14	
17 && 14	
17 || 14	
~17	
!17	
5 > 4 & 0	
17 >> 2	
17 << 2	
0 && 0 || 1	
5 && 0	
5 || 0	

Problem 16 (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 = 107 ;
int    V[3] = {181, 202, 255} ;
int    *p = NULL ;
int    *q = NULL ;
```

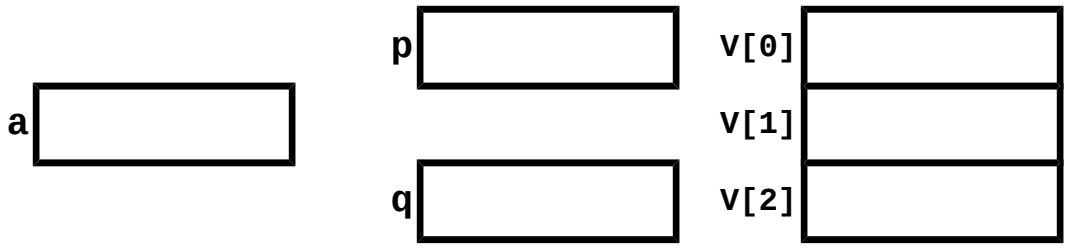
As you might guess, **null** in Java is similar to **NULL** in C. Draw the value **NULL** with a little **X**. Don't ever just leave the pointer variable boxes empty.

Code section @ (the starting point)



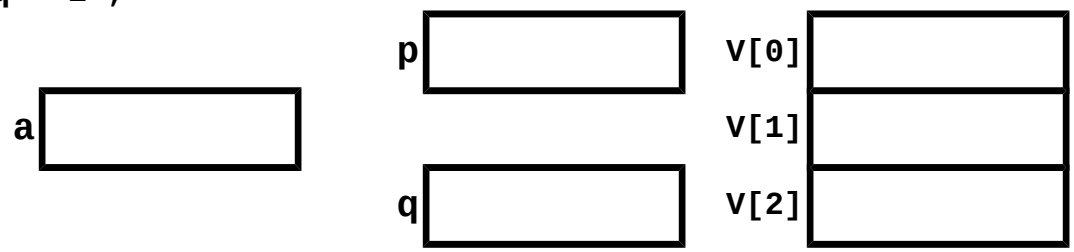
Code section A

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



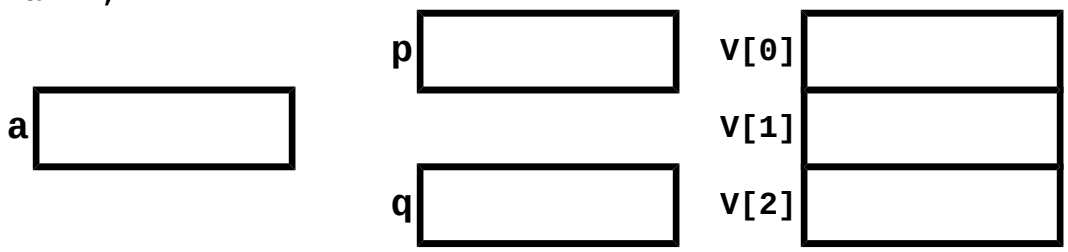
Code section B

```
p = &V[1] ;
q = &V[2] ;
*p = *q + 1 ;
```



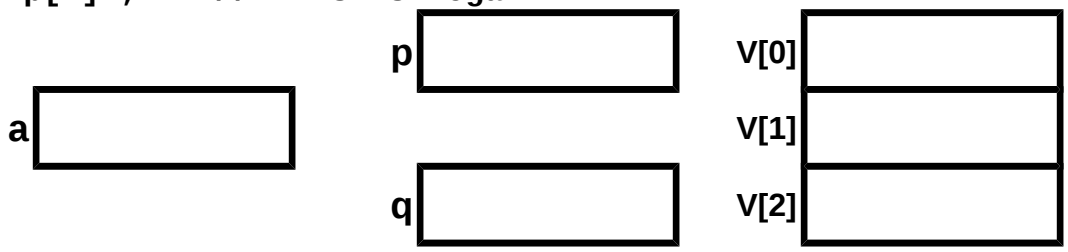
Code section C

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



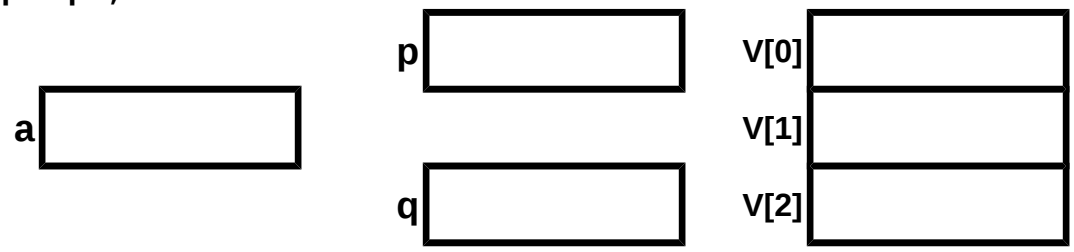
Code section D

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



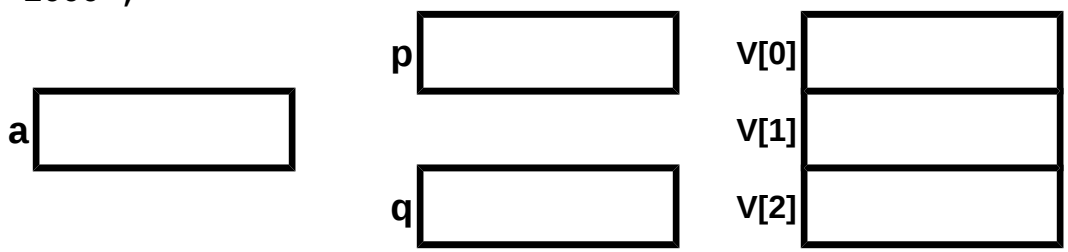
Code section E

```
p = &V[0] ;
q = &V[2] ;
*p = *q – *p ;
*q = q – p ;
```



Code section F

```
p = &V[0] ;
q = &V[1] ;
*p++ = 1000 ;        // same as *(p++) = 1000 ;
*++q = 2000 ;
```



Two very similar programming problems, one in C and the other in MIP32 assembly, remain. The two problem are given here. You will write your answers on the pages that follow.

Problem 17 (10 points)
If the style of the sixth homework, rewrite a section of C code into C code that only uses two control structures:

```
goto label ;
if (expression) goto label ;
```

Do not use the `?:` operator of C and Java to simulate an `if-then-else`.

This specifically means that you can't use the `for`, `while`, `switch`, or even the statement block delimiters `{` and `}`. You can use the `if`, but the conditional expression must be immediately followed by a `goto` statement.

Here is the C program:

```
for (i=0; i<size; ++i) {
  if (0 < V[i]) {
    if (V[i] < 101) {
      satSum = satSum + V[i] ;
    } else {
      satSum = satSum + 100 ;
    }
  }
}
```

Problem 18 (12 points)
Translate the following C subroutine into MIPS32 assembly language.

```
unsigned int saturatedSum(unsigned int V[], int size) {
  unsigned int satSum = 0 ;
  for (int i=0; i<size; ++i) {
    if (0 < V[i]) {
      if (V[i] < 101) {
        satSum = satSum + V[i] ;
      } else {
        satSum = satSum + 100 ;
      }
    }
  }
  return satSum ;
}
```

Remember, the two parameters are passed in `$a0` and `$a1` and the result is returned in `$v0`.

If you want the full 12 points for this question, you need to comment your code.

Problem 17 answer goes on this page

```
for (i=0; i<size; ++i) {
  if (0 < V[i]) {
    if (V[i] < 101) {
      satSum = satSum + V[i] ;
    } else {
      satSum = satSum + 100 ;
    }
  }
}
```

Assume `i`, `size`, `V[]`, and `satSum` have been declared and, if appropriate, initialized.

Problem 18 answer goes on remaining pages

```
unsigned int saturatedSum(unsigned int V[], int size) {
  unsigned int satSum = 0 ;
  for (int i=0; i<size; ++i) {
    if (0 < V[i]) {
      if (V[i] < 101) {
        satSum = satSum + V[i] ;
      } else {
        satSum = satSum + 100 ;
      }
    }
  }
  return satSum ;
}
```

Be sure to comment your code!

```
            .global    saturatedSum
            .ent       saturatedSum
saturatedSum:
```

Problem 18 answer continues

```
unsigned int saturatedSum(unsigned int V[], int size) {
  unsigned int satSum = 0 ;
  for (i=0; i<size; ++i) {
    if (0 < V[i]) {
      if (V[i] < 101) {
        satSum = satSum + V[i] ;
      } else {
        satSum = satSum + 100 ;
      }
    }
  }
  return satSum ;
}
```

Be sure to comment your code!

```
        .end    saturatedSum
        .size   saturatedSum, .-saturatedSum
```