

CS 273
Midterm Exam
March 7, 2008

The following exam is open book and open notes. You may feel free to use whatever additional reference material you wish, but **no electronic aids** are allowed.

Remember the following:

- show your work whenever appropriate. There can be no partial credit unless I see how answers were arrived
- be succinct. You may lose points for facts that, while true, are not relevant to the question at hand

Finally, please note the following instructions. There will be a ten point deduction for failure to comply with them:

- start each problem on a new sheet of paper
- write your Banner number, but not your name, on each sheet of paper you turn in

You have until 10:20 to finish the exam.

1. (10) Using the division algorithm, convert the following numbers from decimal to eight bit binary. If the number is negative, represent it in two's complement.

(a) -37

First, we convert 37 to binary:

<i>Old</i>	<i>New</i>	<i>Bit</i>
37	18	1
18	9	0
9	4	1
4	2	0
2	1	0
1	0	1

giving us 00100101.

Now we negate this by complementing all the bits and incrementing:

Original: 00100101

Complemented: 11011010

Incremented: 11011011

*Giving us a final answer of **11011011***

(b) 44

This time the number is positive, so we can just do the radix conversion:

<i>Old</i>	<i>New</i>	<i>Bit</i>
44	22	0
22	11	0
11	5	1
5	2	1
2	1	0
1	0	1

*The result is **00101100***

Points	Error
-3	Sign-magnitude instead of 2's complement
-1	Less than eight bits
-5	Not division algorithm

2. (10) Using the multiplication algorithm, convert the following signed numbers (so the numbers may be either negative or positive) from eight bit hexadecimal to decimal.

(a) 31

Old	Digit	New
0	3	3
48	1	49

(b) e1

Since the first digit is eight or greater, the most sign bit is 1, so the number is negative. First thing is to convert to positive by complementing the bits and incrementing:

Original: e1

Complemented: 1e

Incremented: 1f

Now we convert to decimal:

Old	Digit	New
0	1	1
16	15	31

So the final result is **-31**.

Points	Error
-3	Completely wrong negation
-1	Octal instead of hex
-5	Not multiplication algorithm
ok	Did b right, forgot to put - in front of answer

3. (15 points) Translate the following machine code instructions into assembly language. You can leave any addresses or other constants as hexadecimal "magic numbers" (all of the numbers in the problem are given in hexadecimal). You will need to make sure you've got the radix and addressing mode notations correct, however.

(a) 1b

aba

(b) c9 12

adcb #\$12

(c) 18 ab 13

adda \$13, y

(d) f0 ff 20

subb \$ff20

(e) 6d 88

tst \$88, x

Points	Error
-1	Each missing #, \$, etc.
-2	Extra prefixes left lying around, other wrong syntax
-3	Completely wrong instruction

4. (15 points) Assuming the following equ's appear in a program,

```
otto    equ 23
anne    equ $6f
margot  equ $f963
edith   equ %10010110
```

translate the following assembly language statements into machine code. For the numbers given in radices other than 16, you'll need to convert (using whatever algorithm you're most comfortable with, not necessarily the ones given in class) the values to hexadecimal.

- (a) ldaa #otto
86 17
- (b) staa anne
97 6f
- (c) adda margot
bb f9 63
- (d) cmpb edith,y
18 e1 96
- (e) aby
18 3a

Points	Error
-1	Radix conversion errors
-2	Wrong addressing mode
-1	Missed '18' prefix

5. (15 points) What will the NZVC condition codes be after the following code fragment is executed? In your answer, give the condition codes following each instruction; if any are unknown based on the information in the problem, say so (at the end of the problem, they will all be known). Will the blt instruction result in taking the branch, or in continuing to the instruction after it?

```
ldaa #83
    N=1, Z=0, V=0, C=?
tab
    N=1, Z=0, V=0, C=?
addb #f6
    N=0, Z=0, V=1, C=1
blt  freddie
    N⊕V=1, so yes
```

Points	Error
-1	Each incorrect condition code
-5	Branch condition

6. (35 points)

Translate the following fragment of high level language pseudocode into HC11 assembler.

```

i = 1
for (a = 4; a != 0; a--)
    if (a > i)
        i = i + i

```

In your code, you may assume that the loop is executed at least once. Also, you should assume the A accumulator is used to hold the variable a, while a memory location is used to hold i. You can assume the proper rmb's, org's, etc. have been written; for this problem, you only need to provide the actual code.

```

        ldaa #1
        staa i
        ldaa #4
loop    cmpa i
        ble noadd
        ldab i
        addb i
        stab i
noadd  deca
        bne loop

```

Points	Error
-5	<i>Infinite loop when a <= i</i>
-2	<i>i++ instead of i+=i</i>
-2	<i>Don't initialize i</i>
-5	<i>Don't init a or i</i>
-5	<i>No label on branch</i>
-2	<i>Wrong branch</i>
-15	<i>handled a OK, but didn't ever change i</i>
-5	<i>Never dec a</i>