

CS 273

Midterm Solutions

October 3, 2005

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. 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 social security number, but not your name, on each sheet of paper you turn in
- 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

You have until 11:20 to finish the exam.

1. (10 points) Assuming the following equ's appear in a program,

```

amanda equ 73
laura  equ $3f
tom    equ $ff62
jim    equ %01101001

```

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 #jim`
`86 69`
- (b) `staa laura`
`97 3f`
- (c) `adda tom`
`bb ff 62`
- (d) `cmpb amanda,x`
`e1 49`

Points	Error
-1	<i>Wrong operand, though correct instruction and mode</i>
-2	<i>Opcode for different instruction</i>
-1	<i>Wrong addressing mode</i>
-2	<i>Forgot radix conversion for operand</i>

2. (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 addressing mode notation correct, however.

- (a) `4f`
`clra`
- (b) `86 12`
`ldaa #$12`

- (c) 9b 43
adda \$43
- (d) f0 ff 20
subb \$ff20
- (e) 6d 88
tst \$88,x

Points	Error
-1	Wrong addressing mode
-2	Wrong instruction

3. (20 points)

(a) Convert the following decimal number into eight bit signed hexadecimal, using the algorithm from class: -47

i. Invert the sign, remembering that it's negative: 47

ii. Perform the conversion to hexadecimal:

Old	New	Digit
47	2	f
2	0	2

which gives 2f.

iii. Invert the bits and add 1: $ff - 2f + 1 = d0 + 1 = d1$.

(b) Add it to the hexadecimal number \$e3 (giving an eight bit result). What is the result of the addition, and what would the condition codes be?

$$\begin{array}{r} d1 \\ + e3 \\ \hline b4 \end{array}$$

N: most significant nybble is greater than or equal to 8, most significant bit is 1, so 1.

Z: it's not 0, so 0.

V: both operands are negative, as is the result so no overflow, so 0.

C: there is a carryout on the addition, so 1.

(c) If the next instruction is a bge, will the branch be taken?

bge is taken if $N \oplus V$ is 0, so not taken.

(d) Translate your signed result from step 3b to decimal, again using the algorithm from class.

i. Note that it's negative (as the most significant bit is 1), so invert it and remember the sign:

$$ff - b4 + 1 = 4b + 1 = 4c$$

ii. Convert to decimal

Old	Digit	New
0	4	4
64	c	76

iii. Put on the minus sign: -76

Points	Error
-1	16 bit
-3	bge wrong
-15	Only part a
-1	Condition code error
-1	Didn't notice negative in part d
-1	Read hex backwards (2f) in part a
-2	Didn't negate in part a
-1	Forgot to add 1 negating
-2	Not algorithm from class

4. (20 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 bge instruction result in taking the branch, or in continuing to the instruction after it?

```

ldaa #7f
adda #60
ldab #2
bge  freddie

```

- (a) *The adda instruction sets all the condition codes, so we don't need to worry about the ldaa. I actually only put that first instruction in to be able to have predictable results after the second one.*
- (b) *The adda instruction calculates $60 + 7f = df$.
The result is negative, so the N bit is 1.
The result is not 0, so the Z bit is 0.
The two operands were positive and the result was negative, so the V bit is 1.
There was no carry-out from the addition, so the C bit is 0.*
- (c) *The ldab instruction sets the N and Z bits according to the operand, clears V, and leaves C alone.
The operand is positive, so N is 0
The operand is non-zero, so Z is 0
V is 0
C is still 0.*
- (d) *The bge instruction decides whether to take the branch depending on whether $N \oplus V$ is 0; it is, so the branch would be taken.*

Points	Error
-2	CC correct, but said not to take branch
-5	Skipped final ldab instruction
-5	Didn't say whether branch is taken
-1	Incorrect CC (each)

5. (35 points) Consider the following assembly code:

```

        ldaa #%00010001
        ldab #4
loop    staa $1004
        adda #%00010001
        decb
        bne  loop

```

- (a) How large is this code, in bytes?
 $2 + 2 + 3 + 2 + 1 + 2 = 12$ bytes.
- (b) The bne instruction uses relative addressing. What will its operand be? I want a specific hexadecimal value for this.
It must branch backward over all the bytes from the staa to the bne. That's $3 + 2 + 1 + 2 = 8$ bytes; the relative address will be -8. In hexadecimal, that's $ff - 08 + 1 = f8$.
- (c) On each iteration of the loop, one or more of the motor lights will be illuminated. What will the lights for each iteration be?
- Motor 1, red.
 - Motor 2, red
 - Motors 1 and 2, red.
 - Motor 3, red.

Points	Error
-5	No relative addressing mode
-10	Completely wrong size with no explanation
-1	-6 bytes
-1	forgot high-order nybble
-2	wrong lights (per cycle)
-5	Multiplied code size by four