

# CS 273

## Midterm Exam

### Solutions

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 Banner ID number, but not your name, on each sheet of paper you turn in

Also, please note the following:

- show your work whenever appropriate. There can be no partial credit unless you show how you derived your answers
- 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. The questions are equally weighted.

1. Suppose you have the following assembly language fragment, defining symbols for a program:

```
RAM      equ 00
EEPROM   equ $f800

          org RAM
i         rmb 5
j         rmb 1

          org EEPROM
k         fcb 2, 3
l         fcb 7
```

*Correct instruction (second digit of op code): 2*

*Correct mode (first digit of op code): 1*

*Correct operand: 2 (off-by-one will be -1)*

*Address instead of symbol: -1*

- (a) Assemble each of the following instructions.

i. `ldaa l`

*l's address is \$f802, so this is extended mode. Looking on page 16, we get*

*B6 f8 02*

ii. `suba #j`

*j's address is \$05. This is immediate mode, so page 19 gives us*

*80 05*

iii. `asl i`

*i's address is \$00. This would be direct mode, but according to page 13 the asl instruction isn't available in direct mode; consequently, it's extended. So we have*

*78 00 00*

(b) Disassemble the following instructions. As on HW2, use symbols in your operands (e.g. use  $i$ , not 0)

i.  $d0\ 00$

*The opcode map on page 8 tells us this is a `sub` instruction on accumulator B, in direct mode. Since  $i$ 's address is 0, this is*

*`subb i`*

ii.  $b8\ f8\ 00$

*The opcode map on page 8 tells us this is `eor`, accumulator a, extended mode. Since  $k$ 's address is  $\$f800$ , this is*

*`eor a, k`*

iii.  $18\ 60\ 00$

*This time, the opcode map on page 8 tells us we need to go to opcode map page 2, which is on page 9 of the reference guide. When we get there, we need to decode opcode 60; this is a `neg` instruction, indexed-y mode. We haven't talked about using symbols in indexed addressing, so I'll accept either*

*`neg 0, y`*

*or*

*`neg i, y`*

2. Convert the following numbers to 2's complement, 8-bit hexadecimal.

*Conversions: 10 points*

*Negation in (b): 10 points*

*Tried to negate by adding  $ff + 1$ : -5*

*Forgot to add +1 on negation: -2*

(a) 39

*39/16 is 2 remainder 7, so we get  $27_{16}$*

(b) -53

*First we convert 53 to hexadecimal. 53/16 is 3 remainder 5, so we have  $35_{16}$ . We need to negate this. Inverting the bits gives  $ca_{16}$ ; adding one makes it  $cb_{16}$ .*

3. Convert the following 2's complement, 8-bit hexadecimal numbers to decimal.

*Conversions: 10 points*

*Negation in (a): 10 points*

(a)  $d4$

*Since the most significant digit is greater than 7, the most significant bit is 1. So, it's negative. First thing is to convert it to positive; inverting the bits gives  $2b_{16}$  and adding one gives  $2c_{16}$ . Converting to decimal we have  $2 \cdot 16 + 12 = 44$ . So, the solution is -44.*

(b)  $3a$

*The most significant digit is less than 8, so it's positive. All we need to do is convert to decimal;  $3 \cdot 16 + 10 = 58$ .*

4. Perform each of the following 8-bit hexadecimal additions, obtaining both the value and condition codes. Then determine whether the branch instruction is taken.

*Six points for the arithmetic, four for the condition codes (1 each), five for the branch.*

*Correct setup for branch instructions but wrong result -2*

(a)  $3a + 77$ , `ble`

*$3a + 77 = b1$ .  $N=1$ ,  $Z=0$ ,  $V=1$ ,  $C=0$ . `ble` depends on  $Z|(N \wedge V)$ ; since this is 0 the branch is not taken. Note that there is an overflow since the two operands have the same sign while the result has the opposite sign.*

(b)  $8c + 63$ , `bhi`

*$8c + 63 = ef$ .  $N=1$ ,  $Z=0$ ,  $V=0$ ,  $C=0$ . `bhi` depends on  $C|Z=0$ ; since  $C$  and  $Z$  are both 0, the branch is taken.*

5. If the following code sequence is executed, what is the final value of any registers and memory locations that are changed by it (including PC and condition codes)?

```
org $f800
ldaa #$5a
asla
staa 04
```

Value in A, PC, and memory address: 5 points each

Condition Codes: -1 each

Bad initial condition on PC: -5

Put initial instruction at f802: -2

No answer at all for A or PC: -10

- (a) `ldaa` loads 5a into the a accumulator. PC advances to f802.
- (b) `asla` shifts the contents of a to the left; the new contents of a are b4 and the PC advances to f803. The C bit gets the old most significant bit of a, so C = 0.
- (c) `staa` stores the contents of a to memory location 4. The PC advances to f805. This instruction sets N and Z according to the value stored in memory so N=1 and Z=0. The instruction clears V, so V = 0. C is unchanged by the instruction, so it still contains 0.

The total changes are: a = b4, PC = f805, location 4 = b4, N = 1, Z = 0, V = 0, C = 0.

6. Assuming variable i is in RAM and variable a is in the A accumulator, translate the following high level language program fragments to assembly code. Assume the variables are signed. These are only fragments, so you *don't* need to give me any of the assembler directives that would be necessary in a complete program (no `equ`, `org`, `end`, etc.)

15 points for each part

Same code as me, but label before `staa`: -2

Wrong branch instruction: -2

Strange permutations on labels: -2 (e.g. `bgt eloop`, but label is `loop`, and indented)

Extra instructions that make it wrong: -5

Wrong addressing mode: -2

Extra instructions that don't matter: OK

- (a) `if (a < i)`  
    `j = a;`
- ```
        cmpa i
        bge noif
        staa j
noif
```
- (b) `do {`  
    `a = a -1`  
    `} while (a != 0);`
- ```
loop
    deca
    bne loop
```