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:

   claudius equ 10
   gertrude equ $13
   polonius equ $f903

   translate the following assembly language statements into machine code.

   (a) ldac #claudius
       86 0a
   (b) stac gertrude
       97 13
   (c) addc #polonius
       Ummm, there is a typographical error here which nobody in the class caught (or at least nobody asked me about during the exam): you can’t have a 16 bit immediate operand on an adda (my intent had been to make this one extended). So this one won’t be graded.
       (d) cmpb claudius, y
           18 e1 0a
       -1 No prefix on (d)
       -1 Wrong addressing mode
       -2 No operand and no clue why
       -1 Didn’t convert operand to hex (or converted operand already hex)
       -2 Opcode for entirely different instruction
2. (10 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)

(a) 1B
   aba
(b) 18 a4 37
   anda $37, y
(c) c1 13
   cmpb #$13
(d) ce f8 03
   ldx #$f803

1. Missing byte of operand
2. Wrong notation for addressing mode
3. Completely wrong instruction

3. (25 points)

(a) Convert the following decimal number into eight bit (signed) hexadecimal: -40

   i. Convert 40 to hexadecimal
      
      | Old | New | Digit |
      |-----|-----|-------|
      | 40  | 2   | 8     |
      | 2   | 0   | 2     |

      giving 28.

   ii. Negate:

      Binary: 0010 1000
      Invert bits: 1101 0111
      Add one: 1101 1000

      So the result is d8.

   -1 Didn’t invert all bits
   -4 Decimal-hex wrong; no idea where it came from
   -2 Inversion just plain wrong
   -1 “13” instead of “d”
   -1 Negated by subtracting from 256, but used 255
   -1 Binary, not hex

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

   \[
   \begin{array}{c}
   \text{d8} \\
   + \text{ce} \\
   \hline
   \text{a6}
   \end{array}
   \]

   \begin{align*}
   N: & \quad 1 \quad \text{(sign bit is 1 so it is negative)} \\
   Z: & \quad 0 \quad \text{(not zero)} \\
   V: & \quad 0 \quad \text{(sign of result is the same as the sign of the operands)} \\
   C: & \quad 1 \quad \text{(there is a carry-out from the addition)}
   \end{align*}

   -1 Kept leading “1” (forgot eight bit addition)
   -3 Didn’t know how to carry
   -1 Condition Codes (each)

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

No. The branch would be taken if \( N=0 \).

-4 Yes
Translate your signed result from step 3b to decimal.

i. The sign bit is 1, so the number is negative. Negate it.

Binary: 1010 0110
Invert bits: 0101 1001
Add 1: 0101 1010

ii. Convert to decimal: $5 \times 16 + 10 = 90$.

iii. Put the minus sign on it: -90

-2 Didn’t recognize negative
-1 Subtract 1 instead of adding 1
-8 Didn’t do it.
-4 Converted wrong number

4. (20 points) How many cycles will it take to execute the following code?
(assume bogus is in RAM).

<table>
<thead>
<tr>
<th>Code</th>
<th>Cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td>ldab #10</td>
<td>2</td>
</tr>
<tr>
<td>loop inca</td>
<td>$10 \times 2 = 20$</td>
</tr>
<tr>
<td>decb</td>
<td>$10 \times 2 = 20$</td>
</tr>
<tr>
<td>bne loop</td>
<td>$10 \times 3 = 30$</td>
</tr>
<tr>
<td>staa bogus</td>
<td>3</td>
</tr>
</tbody>
</table>

75 cycles

-5 Didn’t multiply by loop iterations
-1 Didn’t count an instruction
-2 Got 5 loop iterations
-1 Wrong number of cycles for an instruction
richard = 43;
while (richard > 0) {
    if (richard < kyle)
        kyle = richard + kyle;
    richard = richard - 1;
}
    ldaa #43
    staa richard    * richard = 43;
wloop tst richard    * while (richard > 0) {
    ble out
    ldaa richard    * if (richard < kyle)
    cmpa kyle
    bge skip
    adda kyle    * kyle = kyle + richard;
    staa kyle
    skip dec richard    * richard = richard - 1;
    bra wloop    *
}
out

A number of people tried to write optimized solutions. So, here’s an example of an optimized solution.

ldab #43
ldaa kyle
loop cba
ble skip
aba
skip decb
bgt loop
stab richard
staa kyle

There was a formatting problem with the question that could easily give the impression that the richard = richard-1; line was within the scope of the if. A careful reading would show that it wasn’t, but I’ll accept solutions that thought it was.

-2 ldaa #richard instead of ldaa richard
-2 bgt instead of ble
-2 Didn’t write decremented richard back
-4 No while loop
-0 check for 0 in while loop (instead of negative) - technically wrong, but OK
-2 Put richard+kyle code outside while so it doesn’t participate in loop
-2 Fall through after loop into richard+kyle code
-4 Always compute richard+kyle, but not always richard-1