CSC 258 Assignment 3, Fall 2008

Due by 5:00 p.m., Friday November 14, 2008; no late assignments without written explanation.

1. Write a program in the VELMA assembly language to count the number of '1' bits in a word, using the following algorithm. The result should be in R0 when your program HALTs.

> (i) Start with the number 1.
> (ii) AND this number with the word in question (with "BIT"). If you get a non-zero value, then the low bit is set, so add one to your counter.
> (iii) Logically-left-shift that '1' by one place (with "LSHL 1").
> (iv) Repeat from step ii to examine all of the other bits.

Write your answer in a file named "q1". The last line of the file must be of the form:

```
X: .WORD 123
```

Our test programs will automatically substitute this line with various values for automated testing. The name must be X.

2. What does the following VELMA code do (in English)?

```
     MOV# 2000, R0
XYZ: INC R0
     MOV @R0, R1
     TST R1
     BNE XYZ
     HALT
```

(Note: "`MOV @R0, R1`" is the same as "`MOV (R0), R1`")

Please write your answer in a plain text file named "q2".

3. Here is a VELMA machine language program, in octal.

loc	value
01000	17002000
01001	17103000
01002	17400047
01003	30200000
01004	20000003
01005	06400003
01006	70201010
01007	31200000
01010	31000000
01011	06100000
01012	71101004
01013	77000000

(a) Disassemble this program (convert it to assembly language). You needn't introduce labels. Remember that the registers 0 through 7 are also available as memory locations 0 through 7, respectively. Please produce a file named "q3a" which is a valid VELMA program (beginning

(over)

with ‘‘.ORG 1000’’).

(b) Describe simply what it does, in a file named ‘‘q3b’’.

4. In a file named ‘‘q4’’, write a subroutine in the VELMA assembly language to compute the sum of an array of signed 24-bit numbers (normal VELMA words; i.e. you can use ADD). An example calling sequence to add 50 words starting at location 1000 is:

```
MOV #1000, R0
MOV R0, −(R6)
MOV #50, R0
MOV R0, −(R6)
JSR ROUTINE
INC R6
INC R6
```

You return the computed sum in R0. Your subroutine may thus use R0 as a scratch register, but any other registers used must be pushed and restored.

Your subroutine is required to check for overflow. If none of the addition operations overflows, your subroutine must return the sum in R0, and with the V condition code clear. If any of the addition operations overflow, your subroutine must return with the V condition code set, and in that case it doesn’t matter what the final contents of R0 is. (So there would actually probably be a BVS instruction between the JSR and the INC in the calling sequence above.)

You need not handle the case of zero-sized arrays; the count (second parameter) must be positive. You are not required to handle any kind of incorrect use of your subroutine.

Your code needn’t be especially well-commented but you should explain your use of registers and anything else particularly unclear.

Do not assemble your code; leave it in symbolic form.

Submission:

Your assignment three solutions will be submitted on-line on the CDF computers. Write your answers in five separate files, named ‘‘q1’’, ‘‘q2’’, ‘‘q3a’’, ‘‘q3b’’, and ‘‘q4’’. Make sure that they are all plain text files by displaying them in a terminal window using ‘‘cat’’ (e.g. ‘‘cat q2’’—be particularly careful that your q2 and q3b answers are plain text).

Your files ***must*** have the correct names to be correctly processed by the grading software. Once you are satisfied with your files, you can submit them for grading with the command

```
submit −c csc258h −a a3 q1 q2 q3a q3b q4
```

You can still change your files and resubmit them any time up to the due time. You can check that your assignment has been submitted with the command

```
submit −l −c csc258h −a a3
```

You can submit files individually instead of all at once, and you can resubmit a particular file by using the −f option, e.g.

```
submit −c csc258h −a a3 −f q4
```

This is the only submission method; you do not turn in any paper.