CMPE 12/L

Homework 3

Katerina Chinnappan

kachinna@ucsc.edu

1) (15 pts) Write an LC-3 assembly language program that counts the number of 1s in the value stored in R0 and stores the result in R1. For example, if R0 contains 0001001101110000, then R1 should store the value 6

Answer:

```
;Summary of the registers I will use and their purpose
;R1 register will contain the counter of how many 1's occur
;R2 register will be the loop counter (how many times we must go thru)
;R3 register, I will use this as a temp register to copy whatever is in R0

        .ORIG x3000          ;this is where our program begins
        AND R1, R1, 0        ;clear the registers before using them
        AND R2, R2, 0
        AND R3, R3, 0
        ADD R3, R0, 0        ;copy whatever is in R0 into R3
        ADD R3, R3, 0        ; add 0 to check for the highest bit (neg or pos number?)
        BRZP FOO             ;go to FOO if greater than 0 (so option is 1)
FOO
        ADD R2, R2, -15      ;this is our counter, we need to loop 15 times thru the
                             ;remaining 15 bits
LOOP1
        BR LOOP2             ;if >=, go to loop2
        ADD R1, R1, 1        ;count
LOOP2
        ADD R2, R2, 1        ;increment the counter
        BRn LOOP1            ;if not done, go back, otherwise bye
HALT
.END
```

2) (10 pts) The following program adds the values stored in memory locations A, B and C, and stores the result into memory. There are two errors in the code. For each, describe the error and indicate whether it will be detected at assembly time or run time.

```
1. .ORIG x3000
2. ONE LD R0, A
3. ADD R1, R1, R0
4. TWO LD R0, B
5. ADD R1, R1, R0
6. THREE LD R0, C
7. ADD R1,R1,R0
8. ST R1, SUM
9. HALT
10. A .FILL x0001
11. B .FILL x0002
12. C .FILL x0003
13. D .FILL x0004
14. .END
```

Answer: 1) First mistake is that there is no label for SUM, and that will be detected during assembly time. 2) Second mistake is that R1 isn't cleared or initialized…. So that's detected in the run time.

3) (5 pts) Name some of the advantages of doing I/O through a TRAP routine instead of writing the routine yourself each time you would like your program to perform I/O.

Answer: Basically….we don't have to go thru the difficulty of understanding the details like which register to use for input and output data . Long story short…makes our life easier.

4) (5 pts) 1. How many trap service routines can be implemented in the LC-3? Why? 2. How many accesses to memory are made during the processing of a TRAP instruction?

Answer: 1) trap vector 8 bits wide….so 256 trap routines can be implanted.

2)1 memory access – get the starting address by accessing memory location by extended trap-vector.

5) (20 pts) Consider the following LC-3 assembly language program:

1) .ORIG x3000

2) L1 LEA R1, L1

3) AND R2, R2, 0

4) ADD R2, R2, 2

5) LD R3, P1

6) L2 LDR R0, R1, xC

7) OUT

8) ADD R3, R3, -1

9) BRz GLUE

10) ADD R1, R1, R2

11) BR L2

12) GLUE HALT

13) P1 .FILL xB

14) .STRINGZ “HBoeoakteSmtHaotren!s”

15) .END

1. After this program has run, what binary pattern is stored in memory location x3005?

2. Which instruction (provide a memory address) is executed after instruction x3005 is executed?

3. Which instruction (provide a memory address) is executed prior to instruction x3006

4. What is the output of this program?

Answer: 1) The TRAP instruction OUT is executed with the binary pattern: 1111 0000 0010 0001 in location x3005

2) According to LC3 simulator, the instruction ST R7, x043B is executed with the memory address of x0430.

3) According to LC3 simulator, the instruction RET is executed with the memory address of x0437.

4) The output of the program is “HookemHorns”

6) (30 pts) Perform the following multiplications in 4-bit 2SC and provide both the binary and decimal answers. 1. 5 x 6 2. -6 x 4 3. 2 x 6 4. -3 x -2 5. -8 x 7

Answer: 5 x 6 -→ 5 in binary is 0101 and 6 is 0110 →

```
    0101 (5)
X   0110 (6)
-----------------------------
    0000
   01010
  010100
+ 0000000
---------------------------------
  000 11110 which is 30 in decimal.
```

Answer: -6 x 4 → -6 in binary is 1010 and 4 is 0100 →

```
    1010 (-6)
X   0100(4)
-----------------------------
    0000
   00000
  101000
+ 0000000
---------------------------------
  101000 which is -24 in decimal.
```

Answer: 2 x 6 → 2 in binary is 0020 and 6 in binary is 0110 →

```
    0101 (5)
X   0110 (6)
-----------
    0000
   00100
  001000
+ 0000000
-------------
```

0000 1100 which is 12 in decimal.

Answer: -3 x -2 → instead of multiplying negative….just multiply positive since negative times negative is positive (aka additive inverse). 3 in binary is 0011 and 2 in binary is 0010.

→

```
    0101 (3)
X   0110 (2)
-----------
    0000
   00110
  000000
+ 0000000
-------------
```

1001 000which is -56 in decimal.

Answer: -8 x 7 (I will be using sign extension here because I find it easier)

-8 in binary is 1000 and 7 in binary is 0111

→

1000(-8)

X 0111(7)

1000

10000

100000

\+ 0000000

1111000

11110000

1110000000

---

000 11110 which is 30 in decimal.