

# ECE/CS 252 Fall 2011 Homework 7 (25 points) // Due in Lecture on Nov. 30, 2011 (Wed)

## Solutions

### Problem 1. (2 + 3 + 2 points)

Here is a small program to add numbers starting at address labeled as DATA and the number of numbers to be added is given at address labeled as NUM. The program stores the result at location labeled as RESULT.

Output expected (in memory location RESULT): 6

```

        .ORIG x3000
        AND R0, R0, #0
        LD  R3, NUM
        LEA R1, DATA
        LDR R2, R1, #0
LOOP:   BRz DONE
        ADD R0, R2, R0
        ADD R1, R1, #1
        LDR R2, R1, #0
        ADD R3, R3, #-1
        BR LOOP
RESULT: .FILL x0000
NUM:    .FILL x0004
DATA:   .FILL x0000
        .FILL x0001
        .FILL x0002
        .FILL x0003
DONE:   ST R0, RESULT
        HALT
        .END
```

(a) Write the symbol table created by the assembler on the first pass of the above program.

LABEL	ADDRESS
LOOP	x3004
RESULT	x300A
NUM	x300B
DATA	x300C
DONE	x3010

(b) Once the symbol table is created, the assembler then creates a binary version (.obj) of the program. Convert the above program into machine code.

```
0101 000 000 1 00000 ; AND R0, R0, #0
0010 011 000001001 ; LD R3, NUM ; PC-offset = x300B - x3002 = 0x9
1110 001 000001001 ; LEA R1, DATA; PC-offset = x300C - x3003 = 0x9
0110 010 001 000000 ; LDR R2, R1, #0
0000 0 1 0 000001011 ; BRz DONE; PC-offset = x3010 - x3005 = 0xB
0001 000 010 0 00 000 ; ADD R0, R2, R0
0001 010 010 1 00001 ; ADD R1, R1, #1
0110 010 001 000000 ; LDR R2, R1, #0
0001 011 011 1 11111 ; ADD R3, R3, #-1
0000 1 1 1 111111010 ; BR LOOP; PC-off = x3004 - x300A = -6 = 0x1FA (9-bit)
0011 000 111111001 ; ST R0, RESULT; PC-off = x300A - x3011 = -7 = 0x1F9 (9-bit)
1111 0000 0010 0101 ; TRAP x25 or HALT
```

It is fine if you have written binary for .FILL as well though you were not required to.

(c) The code doesn't function as it should. Your job is to debug and fix it. Please explain why it does not work and what needs to be done in order to make it work as expected. You don't need to turn-in full corrected program.

In the first run of the program, instruction labeled as LOOP i.e. BRz Done will act on condition codes (n, z or p flags) set by instruction above it i.e. LDR R2, R1, #0. Since the first number is 0, the program would branch to DONE and store 0 into RESULT without adding further data.

To fix, BRz DONE should act on how many numbers remain to be added. Initially, that is being stored into R3. Hence, value in R3 should be checked before the Branch instruction. Two possible fixes are:

1. Have a dummy instruction that would affect the condition code without affecting the value in R3:

```
LD R3, NUM
LEA R1, DATA
LDR R2, R1, #0
ADD R3, R3, #0
BRz Done
```

2. Move instruction LD R3, NUM before branch instruction:

```
LEA R1, DATA
LDR R2, R1, #0
LD R3, NUM
BRz Done
```