EE 306 Fall 2008 Problem Set 4, Solutions

1. a. 2^{14} (bytes) $\Rightarrow$ 4096 (locations) $\times$ 32 (bits/location) $= 2^{12} \times 4$ (bytes/location)
$= 2^{14}$ (bytes)

b. 12 (bits) $\Rightarrow$ 4096 (locations) $= 2^{12} \Rightarrow$ 12 (bits)

c. 7 (bits) $\Rightarrow$ $50 < 64 = 2^6 \Rightarrow 6+1 = 7$ (bits); Because we would like to jump forward and backward up to 50 locations away and the PC-relative offset uses 2's complement, there should be 7 bits to specify $-64 \sim 63$.

d. 6 $\Rightarrow$ The incremented PC $= 3+1 = 4$, so $10 - 4 = 6$.

2. a. x4030 $\Rightarrow$ The LDR instruction uses 6-bit 2's complement offsets.
The largest address = x4011 + x1F (31 in decimal) = x4030

b. x4050 $\Rightarrow$ For a 6-bit unsigned integer (zero-extended), the largest number is 63.
The largest address = x4011 + x3F (63 in decimal) = x4050

c. x4011 $\Rightarrow$ The smallest address = x4011 + x00 = x4011

3. (2): 1001 1010 1011 1111
(4): 1001 0111 1011 1111
$\Rightarrow$ From DeMorgan's law, A OR B = NOT((NOT(A) AND (NOT(B))

The complete instructions:
```
NOT R4, R1
NOT R5, R2
AND R6, R4, R5
NOT R3, R6
```

4. The value in R6 = x123F

The single instruction = 1010 1100 0011 1111 (LDI R6, #63)

⇒ The three instructions in assembly language:

```
x3010  LEA R3, #63      ; R3 ← (incremented PC) + x3F = x3050
x3011  LDR R4, R3, #0   ; R4 ← mem[R3+#0] = x70A4
x3012  LDR R6, R4, #0   ; R6 ← mem[R4+#0] = x123F
```

⇒ They can be replace with one LDI instruction:

```
LDI R6, #63   ; R6 ← mem[mem[(incremented PC)+x3F)]] = x123F
```

(1010 1100 0011 1111)

5. The error: x3003 0000 0101 1111 1101 ; BRz #-3, this branch instruction should be "BRp #-3", branching if positive. Because if we want to multiply R1 with R2, we should add R2 to itself as many times as the positive number in R1. So the program should branch back if R1 is still positive.

Modification: x3003 0000 0011 1111 1101 ; BRp #-3

6. 0010 0011 1111 1111 (LD R1, #-1)

⇒ This instruction, ASCII LD R1, ASCII, will load the content at memory location pointed by the label "ASCII", and the content is the instruction itself. The offset, -1, is due to the incremented PC which points to the next memory location.

7. The error: ADD R3, R3, #30, where the offset #30 is beyond the range which can be represented by the 5-bit 2's complement immediate field of the ADD instruction.

How to fix it: use the ADD instruction twice, ADD R3, R3, #15, to add #30 into R3.

This error will be detected when this code is assembled.

8. 9, 10, 11 ⇒ To cause the final value in R2 to be 3, it means that the program should branch back to "LOOP" three times, and at the third time it should branch to "END" before the addition. So there are three possible initial values of R1: 9, 10, 11.

9. a.

Symbol	address
LOOP	x3003
L1	x300A
NEXT	x300B
DONE	x300D
NUMBERS	x300E

b. The program counts how many even and odd numbers in the sequence of non-negative integers, and store the results in R3 and R4, respectively.

Therefore, R3 contains the total number of even non-negative integers, and R4 contains the total number of odd non-negative integers.

10. The program finds out the smallest power of 2 which is larger than or equal to the unsigned integer stored in R1.

11. (a) LDR R3, R1, #0

(b) NOT R4, R4

(c) ADD R4, R4, #1

((b) NOT R3, R3

(c) ADD R3, R3, #1)

12. The instruction "ADD R2, R2, #-1" should be moved between the instruction "ADD R3, R3, R3" and the instruction "BR LOOP".

⇒ Because the instruction "Loop BRz DONE" tends to check if the counter equals zero and if yes the program should stop, the condition code should be set by the instruction "ADD R2, R2, #-1" with R2 being the counter, not by the instruction "ADD R3, R3, R3".

13.

	15				11				7				3			0
x3000:	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0
x3001:	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	1
x3002:	1	0	1	1	0	0	0	0	0	0	0	0	0	0	1	1
x3003:	1	1	1	1	0	0	0	0	0	0	1	0	0	0	0	1
x3004:	1	1	1	1	0	0	0	0	0	0	1	0	0	1	0	1
x3005:	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0
x3006:	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1

⇒ ① The first instruction is LD, and the second value of the MAR trace is x3005, which must be the address calculated by "incremented PC + PC offset" in the LD instruction.

Therefore, x3001 + offset = x3005 ⇒ offset = x0004

② The second instruction is "ADD R0, R0, #1", and the 2nd and 3rd value of the MAR trace are x3001 and x3002, respectively. So these two trace values are just the PC.

③ The third instruction is STI, and the 4th and 5th trace values are x3006 and x4001, respectively. It means that "incremented PC + offset = x3006" and that the value in memory location at x3006 equals x4001.

Therefore, offset = x0003 ; x3006 : x4001 = 0100 0000 0000 0001.

④ The last trace value is x0021, which is in system memory. So the only instruction which can access system memory is TRAP. Therefore, the instruction at x3003 is "TRAP x21". #