Exam #2 Study Suggestions

1. Recall the format of ARM data processing instructions:

   Bits 31:26  111000
   Bit 25      I bit
   Bits 24:21  opcode
   Bit 20      S bit
   Bits 19:16  Rn
   Bits 15:12  Rd
   Bits 7:0    immediate value (when I=1)
   Bits 3:0    Rm (when I=0)

   All other bits will be 0.

The sixteen operation codes are given in the following table:

<table>
<thead>
<tr>
<th>Opcode</th>
<th>Mnemonic</th>
<th>Opcode</th>
<th>Mnemonic</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>and</td>
<td>1000</td>
<td>tst</td>
</tr>
<tr>
<td>0001</td>
<td>eor</td>
<td>1001</td>
<td>teq</td>
</tr>
<tr>
<td>0010</td>
<td>sub</td>
<td>1010</td>
<td>cmp</td>
</tr>
<tr>
<td>0011</td>
<td>rsb</td>
<td>1011</td>
<td>cmn</td>
</tr>
<tr>
<td>0100</td>
<td>add</td>
<td>1100</td>
<td>orr</td>
</tr>
<tr>
<td>0101</td>
<td>adc</td>
<td>1101</td>
<td>mov</td>
</tr>
<tr>
<td>0110</td>
<td>sbc</td>
<td>1110</td>
<td>bic</td>
</tr>
<tr>
<td>0111</td>
<td>rsc</td>
<td>1111</td>
<td>mvn</td>
</tr>
</tbody>
</table>

a) What is the purpose of the I bit?

b) What is the purpose of the S bit?

c) What is the smallest immediate value which can be used?  The largest?

d) Convert each of the ARM machine language instructions shown below (in hexadecimal) into ARM assembly language instructions.

   e086a007
   e096a007
   e286a007
   e1520003
   ela01003


e) Convert each of the ARM assembly language instructions shown below into ARM machine language instructions (in hexadecimal).

   subs  r5, r4, r3
   cmn   r6, #12
   orr   r8, r7, #0xff
   mov   r5, r4
   eor   r11, r10, r9

f) Recall that an assembler is the program which converts assembly language instructions into machine language instructions.  Assume that an ARM assembler is to be extended to accept the following assembly language instruction:

   inc   r5

The purpose of that instruction is to increment the contents of R5 by 1.  Give the equivalent ARM machine language instruction.
2. Consider the following ARM assembly language statements. For each statement which contains a comment, give the requested information after the instruction is executed. Give the value of the register in hexadecimal and the value of the condition code bits (if requested) in binary.

```assembly
ldr     r0, =0xb60000e5
ldr     r1, =0x7c00008d
eor     r4, r0, r1      @ r4: ________________________
adds    r5, r0, r1      @ r5: ________________________
        @ NZCV: ____________
or     r6, r0, r1      @ r6: ________________________
subs    r7, r0, r1      @ r7: ________________________
        @ NZCV: ____________
asr     r8, r0, #8      @ r8: ________________________
lsl     r9, r0, #12     @ r9: ________________________
asr     r10, r1, #16    @ r10: ________________________
lsl     r11, r1, #20    @ r11: ________________________
```

3. Give the ARM assembly language code segment which is equivalent to the following C statements. Assume that all variables are integers, the value of "AAA" is in register r4, and the value of "BBB" is in register r5.

```c
if (AAA >= 200 && BBB < 150)
{
    AAA = AAA - 15;
}
else
{
    BBB = BBB + 25;
}
```

4. Give the ARM assembly language code segment which is equivalent to the following C statements. Assume that all variables are integers, the value of "AAA" is in register r4, and the value of "BBB" is in register r5.

```c
AAA = 1;
while (AAA < 10)
{
    BBB = BBB + AAA;
    AAA++;```
5. Consider the C source code statements shown below.

```c
int test( int*, int );

int X, Y, Z;

int main()
{
    Z = test( &X, 8*Y );
}
```

Complete the ARM assembly language function definition below so that it is equivalent to the C definition of function "main" (shown above).

```
.data
.balign 4
X: .skip 4
Y: .skip 4
Z: .skip 4

.global main
.text
.balign 4
main:
```
6. Consider the ARM assembly language code segment shown below.

```
.data
.balign 4
LocA:
    .byte   0x55, 0x66, 0x77, 0x88, 0x99, 0xAA
    .byte   0xBB, 0xCC, 0xDD, 0xEE, 0xFF, 0x00
    .byte   0x11, 0x22, 0x33, 0x44, 0x55, 0x66
    .byte   0x77, 0x88
LocB:
    .word   0xffffffff, 0xeceeeeee
    .word   0xdedddddd, 0xcddddccc
    .word   0xbbbbbbb
.text
.balign 4
    ldr     r0, =0x3456789A
    ldr     r1, =LocA
    ldr     r2, =LocB
    ldr     r4, [r1, #0]    @ r4:  ________________________
    ldrsb   r5, [r1, #2]    @ r5:  ________________________
    ldrb    r6, [r1, #4]    @ r6:  ________________________
    ldrsh   r7, [r1, #6]    @ r7:  ________________________
    ldrh    r8, [r1, #8]    @ r8:  ________________________
    strb    r0, [r2, #2]
    strh    r0, [r2, #6]
    str     r0, [r2, #12]
```

a) The contents of the 20 bytes of memory starting at "LocA" are shown below (in hexadecimal).

In the spaces provided next to the assembly language instructions, give the four-byte hexadecimal value in each of the indicated registers after the instructions execute.

```
LocA:  55 66 77 88 99 aa bb cc dd ee ff 00 11 22 33 44 55 66 77 88
```

b) The contents of the 20 bytes of memory starting at "LocB" are shown below (in hexadecimal).

In the memory dump below, give the hexadecimal value in each of the indicated memory locations after the instructions execute. You only need to give the value for memory locations which are changed by the instructions.

```
LocB:  ff ff ff ff ee ee ee ee dd dd dd dd cc cc cc cc bb bb bb bb
```
7. Consider the C source code statements shown below.

```c
struct player
{
    char name[30];
    unsigned int goals;
    unsigned int assists;
    unsigned int points;
};

// Function total: calculate points for exactly one player in array
void total( struct player roster[], unsigned int n )
{
    roster[n].points = roster[n].goals + roster[n].assists;
}
```

Complete the ARM assembly language function definition below so that it is equivalent to the C definition of function "total" (shown above).

```asm
.global total
.text
.balign 4

total:
```