Study Questions (Part 3)
Friday March 1, 2019

Covering:

- ARM addressing mode
- ARM branching and data processing instructions encoding/decoding

As a good start, you need to answer the question at the end of Chapter 3 of the textbook (pages 224-227).

1. Question 3.23 at page 225: What is the effect of the following addressing mode?
   STR **r0**,[r2,r3,ROR #3]!

2. Question 3.25 at page 225: What is the binary encoding of the following instructions?
   a. STRB r1,**[r2]**
   b. LDR **r3**,[r4,r5]!
   c. LDR **r3**,[r4],r5
   d. LDR **r3**,[r4,#-6]!

3. Question 3.26 at page 225: What is the effect of LDR **r0**,[r5,r6, LSL r2]?

4. Question 3.27 at page 225: You go for a job as a computer architecture designer. At your interview, you tell the panel that you have some real neat ideas for some cool processors. For example, you have an instruction that can set an individual bit in memory. The format of your instruction is
   BITSET **r0**,r1,[(r2),r3,r4 LSR r5, #n]
   This instruction sets the bit in r0 located r1 bits from the word whose address is given by the contents of the memory location pointed at by r2, plus the contents of register r3, plus the contents of register r4 shifted left by register r5, plus the constant n. You don't get the job. Why?

5. Question 3.28 at page 225: What effective address is generated by LDR **r0**,[r2,-r3, LSL #1]?

6. Question 3.29 at page 225: Some machines have a find-first-one instruction that counts the location of the first bit set to 1 within the word. Write an ARM sequence of instructions that takes the word in r0 and puts the locations of the first bit set to 1 in r1. Count from the left so that if bit 31 is set, the value returned should be 0. If bit 0 is set, the value returned is 31. If no bit is set, the value returned should be 32.

7. Question 3.32 at page 225: Assume that r2 contains the initial value 00001000_{16}. Explain the effect of each of the following six instructions, and give the value in r2 after each instruction executes
   a. STR r1,**[r2]**
   b. STR r1,**[r2, #8]**
   c. STR r1,**[r2, #8]!**
   d. STR r1,**[r2], #8**
   e. STR r1,**[r2, r0, LSL #8]**

8. Question 3.60 at page 227: A computer has three eight-element vectors in memory, Va, Vb, and Vc. Each element of a vector is a 32-bit word. Write the code to calculate all elements of Vc if the i^{th} element is given by
   $Vc_i = ½ (Va_i + Vb_i)$

9. ARM supports many addressing modes, including immediate addressing, register-to-register, address register indirect, address register indirect with offset, address register indirect with index, auto-indexing pre-indexed , auto-indexing post-indexed , and program counter relative.
   Give an ARM instruction example for each addressing mode.
10. Register r10 contains 0x10000000. Beginning at that address there are four integers in a row (4 bytes each). Write ONE ARM instruction that loads the last integer into register r7.
11. Register r10 contains the address 0x10000100. Write ONE ARM instruction that stores the content of r7 into the four bytes that precede this address.
12. Write only one ARM instruction that adds the value 0x3EC0000 to the value in register r6.
    Encode this ARM assembly instruction to ARM machine language code.
13. Beginning at address 0x100, there are four integer numbers in a row (4 bytes each), i.e., these numbers are at addresses 0x100, 0x104, 0x108, and 0x10C. Assume that register r0 contains 0x100. Write one ARM instruction that loads the value of the last integer (i.e., the 4th integer at location 0x10C) into register r1.
14. Write a suitable ARM code to implement the following code.

```
int total;
int i;

total = 0;
for (i = 10; i > 0; i--)
{
    total += i;
}
```

15. Explain what this fragment of code does. What are the values of registers at the end of the execution?

```
        MOV  R0, #0         ; R0 accumulates total
        MOV  R1, #10        ; R1 counts from 10 down to 1
again   ADD  R0, R0, R1
        SUBS R1, R1, #1
        BNE  again
halt    B    halt           ; infinite loop to stop computation
```

16. Write a suitable ARM code to implement the following code.

```
    a = 40;
    b = 25;
    while (a != b) {
        if (a > b) a -= b;
        else       b -= a;
    }
```

17. Explain what this fragment of code does. What are the values of registers at the end of the execution?

```
        MOV R0, #40      ; R0 is a
        MOV R1, #25      ; R1 is b
again   CMP R0, R1
        SUBGT R0, R0, R1
        SUBLT R1, R1, R0
        BNE again
halt    B halt
```

18. Write a suitable ARM code to implement the following code.

```
iters ← 0
while n ≠ 1:
  iters ← iters + 1
  if n is odd:
    n ← 3 × n + 1
  else:
    n ← n / 2
```

19. Explain what this fragment of code does. What are the values of registers at the end of the execution?

```
        MOV  R0, #5          ; R0 is current number
        MOV  R1, #0          ; R1 is count of number of iterations
again   ADD  R1, R1, #1      ; increment number of iterations
        TST  R0, #1          ; test whether R0 is odd
        BEQ  even
        ADD  R0, R0, R0, LSL #1 ; if odd, set R0 = R0 + (R0 << 1) + 1
        ADD  R0, R0, #1      ; and repeat (guaranteed R0 > 1)
        B    again
even    MOV  R0, R0, ASR #1 ; if even, set R0 = R0 >> 1
        SUBS R7, R0, #1      ; and repeat if R0 != 1
        BNE  again
halt    B    halt            ; infinite loop to stop computation
```

20. What are the values of r0, r1, r2, and r3 *(in hexadecimal)* after executing the following program?

```
      AREA prog, CODE, READWRITE
      ENTRY
      LDR  r0,=AAA1
      LDR  r1,=0x224488
      LDRB r2,[r0,#1]
      LDRB r3,[r0,#2]
Loop  B    Loop
AAA1  DCB  0x10, 0x20, 0x30, 0x40
      END
```

21. What are the values of r0, r1, r2, and r3 *(in hexadecimal)* after executing the following program?

```
      AREA prog, CODE, READWRITE
      ENTRY
      LDR  r0,=AAA1
      LDR  r1,=0x224488
      STR  r1,[r0]
      LDRB r2,[r0,#1]
      LDRB r3,[r0,#2]
Loop  B    Loop
AAA1  DCB  0x10, 0x20, 0x30, 0x40
      END
```

22. What are the values of r0, r2, and r3 *(in hexadecimal)* after executing the following program?

```
      AREA prog, CODE, READWRITE
      ENTRY
      LDR  r0,=AAA1
      LDRB r2,[r0,#4]
      LDRB r3,[r0,#8]
Loop  B    Loop
AAA0  DCB  0x10, 0x20, 0x30, 0x40
AAA1  DCB  0x50, 0x60, 0x70, 0x80
AAA2  DCB  0x90, 0xA0, 0xB0, 0xC0, 0xD0, 0xE0, 0xF0, 0xFF
      END
```

23. What are the values of r0, r2, and r3 *(in hexadecimal)* after executing the following program?

```
        AREA prog, CODE, READWRITE
        ENTRY
        LDR r0,=AAA1
        LDR r2,[r0,#4]
        LDR r3,[r0,#8]
Loop    B    Loop
AAA0    DCB  0x10, 0x20, 0x30, 0x40
AAA1    DCB  0x50, 0x60, 0x70, 0x80
AAA2    DCB  0x90, 0xA0, 0xB0, 0xC0, 0xD0, 0xE0, 0xF0, 0xFF
        END
```

24. What are the values of r0, r1, r2, r3, r4, r5, and r6 *(in hexadecimal)* after executing the following program?

```
      AREA prog, CODE, READONLY
      ENTRY
      MOV  r0, #0xC
      MOV  r1, #0x1
      MOV  r2, #0x8
      MOV  r3, #0x30
      LDRB r4,[r3, -r1, LSL #3]
      LDRB r5,[r3],-r2, ASR #2
      LDRB r6,[r2,  r1, ROR #27]!
      ADD  r1, r0,  r1, LSL r0
Loop  B    Loop
      DCB  0x11, 0x22, 0x33, 0x44, 0x55
      DCB  0x66, 0x77, 0x88, 0x99, 0xAA
      DCB  0xBB, 0xCC, 0xDD, 0xEE, 0xFF
      END
```

25. What are the values of r0, r1, r2, r3, r4, r5, and r6 *(in hexadecimal)* after executing the following program?

```
      AREA prog, CODE, READONLY
      ENTRY
      MOV  r0, #0xC
      MOV  r1, #0x1
      MOV  r2, #0x8
      MOV  r3, #0x30
      LDR r4,[r3, -r1, LSL #3]
      LDR r5,[r3],-r2, ASR #2
      LDR r6,[r2,  r1, ROR #27]!
      ADD  r1, r0,  r1, LSL r0
Loop  B    Loop
      DCD  0x11, 0x22, 0x33, 0x44, 0x55
      DCD  0x66, 0x77, 0x88, 0x99, 0xAA
      DCD  0xBB, 0xCC, 0xDD, 0xEE, 0xFF
      END
```

26. Encode the following ARM assembly program to ARM machine language codes.

```
    AREA branching, CODE, READONLY
    ENTRY
A   ANDEQ r0,r0,r0
B   ANDEQ r0,r0,r0
C   ANDEQ r0,r0,r0
    BEQ   A
D   ANDEQ r0,r0,r0
E   ANDEQ r0,r0,r0
F   ANDEQ r0,r0,r0
    END
```

27. Decode the following ARM machine language codes to ARM assembly program.

```
0x00000000
0x00000000
0x00000000
0x0AFFFFFB
0x00000000
0x00000000
0x00000000
```

28. Encode the following ARM assembly program to ARM machine language codes.

```
    AREA branching, CODE, READONLY
    ENTRY
A   ANDEQ r0,r0,r0
B   ANDEQ r0,r0,r0
C   ANDEQ r0,r0,r0
    BEQ   F
D   ANDEQ r0,r0,r0
E   ANDEQ r0,r0,r0
F   ANDEQ r0,r0,r0
    END
```

29. Decode the following ARM machine language codes to ARM assembly program.

```
0x00000000
0x00000000
0x00000000
0x0A000001
0x00000000
0x00000000
0x00000000
```

30. Encode the following ARM assembly program to ARM machine language codes.

```
    AREA branching, CODE, READONLY
    ENTRY
A   ANDEQ r0,r0,r0
B   ANDEQ r0,r0,r0
C   ANDEQ r0,r0,r0
    BLEQ  A
D   ANDEQ r0,r0,r0
E   ANDEQ r0,r0,r0
F   ANDEQ r0,r0,r0
    END
```

31. Decode the following ARM machine language codes to ARM assembly program.

```
0x00000000
0x00000000
0x00000000
0x0BFFFFFB
0x00000000
0x00000000
0x00000000
```

32. Encode the following ARM assembly program to ARM machine language codes.

```
    AREA branching, CODE, READONLY
    ENTRY
A   ANDEQ r0,r0,r0
B   ANDEQ r0,r0,r0
C   ANDEQ r0,r0,r0
    BLEQ  F
D   ANDEQ r0,r0,r0
E   ANDEQ r0,r0,r0
F   ANDEQ r0,r0,r0
    END
```

33. Decode the following ARM machine language codes to ARM assembly program.

```
0x00000000
0x00000000
0x00000000
0x0B000001
0x00000000
0x00000000
0x00000000
```

34. Encode the following ARM assembly instruction to ARM machine language code.

```
ADC r0,r1,#0x100
```

35. Decode the following ARM machine language code to ARM assembly instruction.

```
0xE2A10C01
```

36. Encode the following ARM assembly instruction to ARM machine language code.

```
ADC r1,r2,#0xFC000003
```

37. Decode the following ARM machine language code to ARM assembly instruction.

```
0xE2A213FF
```

38. Encode the following ARM assembly instruction to ARM machine language code.

```
ADDS r0,r1,r2
```

39. Decode the following ARM machine language code to ARM assembly instruction.

```
0xE0910002
```

40. Encode the following ARM assembly instruction to ARM machine language code.

```
ANDS r2,r3,r4,LSL #4
```

41. Decode the following ARM machine language code to ARM assembly instruction.

```
0xE0132204
```

42. Encode the following ARM assembly instruction to ARM machine language code.

```
EORS r3,r4,r5,LSR #5
```

43. Decode the following ARM machine language code to ARM assembly instruction.

```
0xE03432A5
```

44. Encode the following ARM assembly instruction to ARM machine language code.

```
ORRS r4,r5,r6,ASR #6
```

45. Decode the following ARM machine language code to ARM assembly instruction.
    `0xE1954346`

46. Encode the following ARM assembly instruction to ARM machine language code.
    `RSBS **r5**,r6,r7,ROR #7`

47. Decode the following ARM machine language code to ARM assembly instruction.
    `0xE07653E7`

48. Encode the following ARM assembly instruction to ARM machine language code.
    `SUBS **r6**,r7,r8,RRX`

49. Decode the following ARM machine language code to ARM assembly instruction.
    `0xE0576068`

50. Encode the following ARM assembly instruction to ARM machine language code.
    `ANDEQ **r7**,r8,r9,LSL r6`

51. Decode the following ARM machine language code to ARM assembly instruction.
    `0x00087619`

52. Encode the following ARM assembly instruction to ARM machine language code.
    `EORNE **r8**,r9,r10,LSR r7`

53. Decode the following ARM machine language code to ARM assembly instruction.
    `0x1029873A`

54. Encode the following ARM assembly instruction to ARM machine language code.
    `ORRCS **r9**,r10,r11,ASR r8`

55. Decode the following ARM machine language code to ARM assembly instruction.
    `0x218A985B`

56. Encode the following ARM assembly instruction to ARM machine language code.
    `RSBCC **r10**,r11,r12,ROR r9`

57. Decode the following ARM machine language code to ARM assembly instruction.
    `0x306BA97C`

58. Encode the following ARM assembly instruction to ARM machine language code.
    `MOVMI **r1**,r2`

59. Decode the following ARM machine language code to ARM assembly instruction.
    `0x41A01002`

60. Encode the following ARM assembly instruction to ARM machine language code.
    `MOVPL **r2**,#0xEE`

61. Decode the following ARM machine language code to ARM assembly instruction.
    `0x53A020EE`

62. Encode the following ARM assembly instruction to ARM machine language code.
    `MOVVS **r3**,#0xEE00`

63. Decode the following ARM machine language code to ARM assembly instruction.
    `0x63A03CEE`

64. Encode the following ARM assembly instruction to ARM machine language code.
    `MVNVC **r4**,r5,ASR #6`

65. Decode the following ARM machine language code to ARM assembly instruction.
    `0x71E04345`

66. Encode the following ARM assembly instruction to ARM machine language code.
    `MVNHI **r5**,r6,ROR r7`

67. Decode the following ARM machine language code to ARM assembly instruction.
    `0x81E05776`

68. Encode the following ARM assembly instruction to ARM machine language code.
    `MVNLS **r6**,r7,RRX`

69. Decode the following ARM machine language code to ARM assembly instruction.
    `0x91E06067`

70. Encode the following ARM assembly instruction to ARM machine language code.
    `NEGEQS **r1**,r2`

71. Decode the following ARM machine language code to ARM assembly instruction.
    `0x02721000`

72. Encode the following ARM assembly instruction to ARM machine language code.
    `CMPGE r0,r1`

73. Decode the following ARM machine language code to ARM assembly instruction.
    `0xA1500001`

74. Encode the following ARM assembly instruction to ARM machine language code.
    `CMNLT r0,#0xFF`

75. Decode the following ARM machine language code to ARM assembly instruction.
    `0xB37000FF`

76. Encode the following ARM assembly instruction to ARM machine language code.
    `CMNGT r1,#0xFF00`

77. Decode the following ARM machine language code to ARM assembly instruction.
    `0xC3710CFF`

78. Encode the following ARM assembly instruction to ARM machine language code.
    `CMPLE r2,r3,LSL#8`

79. Decode the following ARM machine language code to ARM assembly instruction.
    `0xD1520403`

80. Encode the following ARM assembly instruction to ARM machine language code.
    `CMPAL r3,r4,LSL r5`

81. Decode the following ARM machine language code to ARM assembly instruction.
    `0xE1530514`

82. Encode the following ARM assembly instruction to ARM machine language code.
    `TEQ r4,r5,LSR #9`

83. Decode the following ARM machine language code to ARM assembly instruction.
    `0xE13404A5`

84. Encode the following ARM assembly instruction to ARM machine language code.
    `TEQ r5,r6,LSR r7`

85. Decode the following ARM machine language code to ARM assembly instruction.
    `0xE1350736`

86. Encode the following ARM assembly instruction to ARM machine language code.
    `TST r6,r7,ASR #0xA`

87. Decode the following ARM machine language code to ARM assembly instruction.
    `0xE1160547`

88. Encode the following ARM assembly instruction to ARM machine language code.
    `TST r7,r8,ASR r9`

89. Decode the following ARM machine language code to ARM assembly instruction.
    `0xE1170958`

90. Encode the following ARM assembly instruction to ARM machine language code.
    `CMN r8,r9,ROR #0xB`

91. Decode the following ARM machine language code to ARM assembly instruction.
    `0xE17805E9`

92. Encode the following ARM assembly instruction to ARM machine language code.
    `CMN r9,r10,ROR r11`

93. Decode the following ARM machine language code to ARM assembly instruction.
    `0xE1790B7A`

94. Encode the following ARM assembly instruction to ARM machine language code.
    `CMP r10,r11,RRX`

95. Decode the following ARM machine language code to ARM assembly instruction.
    `0xE15A006B`