Computer Architecture Assignment #1

Your Student Number: 2020320132

Name: 조민규

<<Notice>>

- Please edit the title of this document correctly.
    - (ex. CA1_2023012345_Tom-Cruise OR CA1_2023012345_홍길동)
- Please write your information(Student number and name) correctly.
- You can write your answers in English or Korean.
- Don't change the layout(colored in Black) of this document. (Please edit just blue-colored part.)
- Before submitting, don't forget to convert the file to a PDF format.

Address 000 | Instruction EA000006 (Example)

a) Change to binary format: 1110 1010 0000 0000 0000 0000 0000 0110
b) Write assembly code: B #8;
c) Describe why you wrote the assembly code like above:
    a. Type of instruction: According to the figure A3-1 in ARM manual, 'Branch and branch with link' is only one instruction set encoding whose values at [25:27] bit is 101. So, I can figure out this instruction is branch instruction.
    b. Operation – Condition Field: According to the A4.1.5(Page A4-10), there is the detail of the branch instruction. 'Operation' part of the instruction said that I should check the condition is passed first. The condition field of this instruction is 1110 and it means the instruction can operate unconditionally.
    c. Operation – L: According to page A4-10, branch instruction branches without storing a return address when L is omitted. In the case of this instruction, it doesn't need to store any return address because the L bit is 0.
    d. Operation – Target Address: According to page A4-10 in ARM manual, the target address is calculated like below.
        i. First, the result of sign-extending the 24-bit signed immediate to 30 bits is 00 0000 0000 0000 0000 0000 0000 0000 0110. (Because the signed immediate is 0000 0000 0000 0000 0000 0110 here.)
        ii. Then, get 0000 0000 0000 0000 0000 0000 0001 1000 by shifting the result left two bits.
        iii. Because the address of this instruction is 0, the content of PC will be 0 + 8 bytes. So, the target address will be (0+8) + 24 = 32(bytes). It means after the operation of this instruction, PC will be move to 32/4 = 8.
        iv. Therefore, I can write the assembly code of this instruction like 'B #8;' because the syntax of branch instruction is 'B{L}{cond} <target_address>'.

d) What is the meaning of the instruction? : The instruction means ‘branch to address 8’.

Address 001 | Instruction EAFFFFFE

a) Change to binary format: 1110_1010_1111_1111_1111_1111_1111_1110
b) Write assembly code: B#1
c) Describe why you wrote the assembly code like above:
A3.1(p.110)에서 [27:25]가 101 이면 branch 명령어이다. 따라서 이 명령어는 branch 명령어임을 알 수 있다.
또 [24]인 L 비트가 0 이므로 돌아올 곳을 R14 에 저장하는 BL 명령어가 아닌 B 명령어 임을 알 수 있다.(p.160)
또 cond 비트인 [31:28]이 1110 이므로 (p.112)에 따라 항상 분기하라는 명령어이다. 따라서 BAL 명령어임을 알 수 있다.
또 하위 24 비트[23:0]은 offset 으로 다음 과정을 거쳐 pc 에 더해진다.
1. 30bit 로 signed extension 한다. -> msb 가 1 이므로 1 을 채운다.
->11/1111/1111/1111/1111/1111/1111/1110
2. 2bit left shift 로 singed 32bit 로 만든다.-> 1111/1111/1111/1111/1111/1111/1111/1000
3. 이 값을 현재 명령어에 8 이 더해진 값을 가진 pc 에 더한다.
이 32bit 값은 2 의 보수를 이용하여 쓰였기 때문에 10 진수로 변환하면 -8 임을 알 수 있다.
따라서 이 명령어의 주소는 (1*4)=4 이므로 현재 pc 값은 4+8 인 12 이다. 따라서 target_address 는 12-8=4 이고 이를 4 로 나누면 1 이다. 따라서 이 명령어는 B #1 이다.
d) What is the meaning of the instruction? : 1 번째 명령어(address001)를 실행하라는 명령어로 무한루프이다.

Address 002 | Instruction EA0000A7

a) Change to binary format: 1110_1010_0000_0000_0000_0000_1010_0111
b) Write assembly code: B#171
c) Describe why you wrote the assembly code like above:
A3.1(p.110)에서 [27:25]가 101 이면 branch 명령어이다. 따라서 이 명령어는 branch 명령어임을 알 수 있다.
또 [24]인 L 비트가 0 이므로 돌아올 곳을 R14 에 저장하는 BL 명령어가 아닌 B 명령어 임을 알 수 있다.(p.160)
또 cond 비트인 [31:28]이 1110 이므로 (p.112)에 따라 항상 분기하라는 명령어이다. 따라서 BAL 명령어임을 알 수 있다.
또 하위 24 비트[23:0]은 offset 으로 다음 과정을 거쳐 pc 에 더해진다.
1. 30bit 로 signed extension 한다. -> msb 가 0 이므로 0 을 채운다.
->00/0000/0000/0000/0000/0000/1010/0111
2. 2bit left shift 로 singed 32bit 로 만든다.-> 0000/0000/0000/0000/0000/0010/1001/1100
3. 이 값을 현재 명령어에 8 이 더해진 값을 가진 pc 에 더한다.
이 32bit 값은 10 진수로 변환하면 668 이다.

따라서 이 명령어의 주소는 (2*4)=8 이므로 현재 pc 값은 8+8 인 16 이다. 따라서 target_address 는 16+668=684 이고 이를 4 로 나누면 171 이다. 따라서 이 명령어는 B #171 이다.

d) What is the meaning of the instruction? : 171 번째 명령어(address 0AB)로 가라는 뜻

Address 003~005 | Instruction EAFFFFFE

a) Change to binary format: 1110/1010/1111/1111/1111/1111/1111/1110
b) Write assembly code: B#3, B#4, B#5
c) Describe why you wrote the assembly code like above:

A3.1(p.110)에서 [27:25]가 101 이면 branch 명령어이다. 따라서 이 명령어는 branch 명령어임을 알 수 있다.
또 [24]인 L 비트가 0 이므로 돌아올 곳을 R14 에 저장하는 BL 명령어가 아닌 B 명령어 임을 알 수 있다.(p.160)
또 cond 비트인 [31:28]이 1110 이므로 (p.112)에 따라 항상 분기하라는 명령어이다. 따라서 BAL 명령어 임을 알 수 있다.
또 하위 24 비트[23:0]은 offset 으로 다음 과정을 거쳐 pc 에 더해진다.
1. 30bit 로 signed extension 한다. -> msb 가 1 이므로 1 을 채운다.
->11/1111/1111/1111/1111/1111/1111/1110
2. 2bit left shift 로 singed 32bit 로 만든다.-> 1111/1111/1111/1111/1111/1111/1111/1000
3. 이 값을 현재 명령어에 8 이 더해진 값을 가진 pc 에 더한다.
이 32bit 값은 2 의 보수를 이용하여 쓰였기 때문에 10 진수로 변환하면 -8 임을 알 수 있다.
따라서 이 명령어의 주소는 (3*4)=12 이므로 현재 pc 값은 12+8 인 20 이다. 따라서 target_address 는 20-8=12 이고 이를 4 로 나누면 3 이다. 따라서 이 명령어는 B #3 이다.
여기서 address 003, 004, 005 로 pc 값이 4 씩 증가하므로 남은 두 명령어는 B#4, B#5 이다.

d) What is the meaning of the instruction? : 이들 모두 자기 자신으로 분기하라는 명령어 이므로 무한루프이다.

Address 006 | Instruction EA0000A4

a) Change to binary format: 1110/1010/0000/0000/0000/0000/1010/0100
b) Write assembly code: B#172
c) Describe why you wrote the assembly code like above:

A3.1(p.110)에서 [27:25]가 101 이면 branch 명령어이다. 따라서 이 명령어는 branch 명령어임을 알 수 있다.
또 [24]인 L 비트가 0 이므로 돌아올 곳을 R14 에 저장하는 BL 명령어가 아닌 B 명령어 임을 알 수 있다.(p.160)
또 cond 비트인 [31:28]이 1110 이므로 (p.112)에 따라 항상 분기하라는 명령어이다. 따라서 BAL 명령어 임을 알 수 있다.
또 하위 24 비트[23:0]은 offset 으로 다음 과정을 거쳐 pc 에 더해진다.

1. 30bit 로 signed extension 한다. -> msb 가 0 이므로 0 을 채운다.
->00/0000/0000/0000/0000/0000/1010/0100
2. 2bit left shift 로 singed 32bit 로 만든다.-> 0000/0000/0000/0000/0000/0010/1001/0000
3. 이 값을 현재 명령어에 8 이 더해진 값을 가진 pc 에 더한다.
이 32bit 값은 10 진수로 변환하면 656 이다.
따라서 이 명령어의 주소는 (6*4)=24 이므로 현재 pc 값은 24+8 인 32 이다. 따라서 target_address 는 32+656=688 이고 이를 4 로 나누면 172 이다. 따라서 이 명령어는 B #172 이다.

d) What is the meaning of the instruction? : 172 번째 명령어(address 0AC)로 분기하라는 명령어

Address 007 | Instruction EAFFFFFE

a) Change to binary format: 1110/1010/1111/1111/1111/1111/1111/1110
b) Write assembly code: B#7
c) Describe why you wrote the assembly code like above:
A3.1(p.110)에서 [27:25]가 101 이면 branch 명령어이다. 따라서 이 명령어는 branch 명령어임을 알 수 있다.
또 [24]인 L 비트가 0 이므로 돌아올 곳을 R14 에 저장하는 BL 명령어가 아닌 B 명령어 임을 알 수 있다.(p.160)
또 cond 비트인 [31:28]이 1110 이므로 (p.112)에 따라 항상 분기하라는 명령어이다. 따라서 BAL 명령어 임을 알 수 있다.
또 하위 24 비트[23:0]은 offset 으로 다음 과정을 거쳐 pc 에 더해진다.
1. 30bit 로 signed extension 한다. -> msb 가 1 이므로 1 을 채운다.
->11/1111/1111/1111/1111/1111/1111/1110
2. 2bit left shift 로 singed 32bit 로 만든다.-> 1111/1111/1111/1111/1111/1111/1111/1000
3. 이 값을 현재 명령어에 8 이 더해진 값을 가진 pc 에 더한다.
이 32bit 값은 2 의 보수를 이용하여 쓰였기 때문에 10 진수로 변환하면 -8 임을 알 수 있다.
따라서 이 명령어의 주소는 (7*4)=28 이므로 현재 pc 값은 28+8 인 36 이다. 따라서 target_address 는 36-8=28 이고 이를 4 로 나누면 7 이다. 따라서 이 명령어는 B #7 이다.

d) What is the meaning of the instruction? : 7 번째 명령어인 자기 자신으로 분기, 무한루프

Address 008 | Instruction E59F2EC8

a) Change to binary format: 1110/0101/1001/1111/0010/1110/1100/1000
b) Write assembly code: LDR r2 [pc, #3784]
c) Describe why you wrote the assembly code like above: YOUR ANSWER
A3.1(p.110)에서 [27:25]가 010 이면 load/store immediate 명령어이다. 또 [20]인 L 비트가 1 이므로 load 명령어이다.(p.460).

U 비트 [23]은 1, B 비트[22]는 0 이므로 offset_12[11:0]을 Rn[19:16]인 r15 에 더해 address 를 구한다.
address=r15 + offset_12 = r15 + 3784 이다.
또 cond field 인 [31:28]이 1110 이므로 (p.112)에 따라 항상 실행하라는 명령어이다.
단, 실제 실행시에는 PC 에 8 을 더한 후 3784 를 더해서 address 가 계산된다.
또 Rd[15:12]는 0010 으로 r2 를 가리킨다. 따라서 이 명령어는 r2 에 pc+3784 에 있는 값을 load 하라는 명령어로 ldr r2, [pc, #3784] 이다.

d) What is the meaning of the instruction? : pc 에 3784 를 더한 주소를 가진 메모리에 들어있는 값을 r2 레지스터에 저장한다.

Address 009 | Instruction E3A00040

a) Change to binary format: 1110_001_1101_0_0000_0000_0000_01000000
b) Write assembly code: MOV r0, #64
c) Describe why you wrote the assembly code like above:
A3.1(p.110)에서 [27:25]가 001 이고 [23:23]이 10 이 아니면 data processing immediate 명령어이다.
또 A3.4(p.115)에서 opcode 가 1101 이면 MOV 명령어이므로 이 명령어는 MOV 명령어이다.
또 S 비트[20]은 cpsr 을 업데이트 할지 정하는 비트인데 0 으로 설정되어 있으므로 cpsr 을 업데이트 하지 않는다.(p.218)
또 Rd field[15:12]가 0000 으로 r0 를 가리킨다.
또 cond field 인 [31:28]이 1110 이므로 (p.112)에 따라 항상 실행하라는 명령어이다.
Rotate_imm field[11:8]에 2 를 곱한 만큼 immed_8 field[7:0]을 right rotate 해서 rd 에 들어갈 값을 정하는데 여기선 [11:8]이 0 이므로 10 진수 64 를 나타내는[7:0]이 그대로 들어간다.
따라서 이 명령어는 MOV r0, #64 이다.
d) What is the meaning of the instruction? R0 레지스터에 10 진수 64 를 저장하는 명령어이다.

Address 00A | Instruction E5820010

a) Change to binary format: 1110_0101_1_0_0_0_0010_0000_000000010000
b) Write assembly code: STR r0, [r2, #16]
c) Describe why you wrote the assembly code like above:
A3.1(p.110)에서 [27:25]가 010 이면 load/store immediate 명령어이다. 또 [20]인 L 비트가 0 이므로 store 명령어이다.(p.460).
U 비트 [23]은 1, B 비트[22]는 0 이므로 offset_12[11:0]을 Rn[19:16]인 r2 에 더해 address 를 구한다.
address=r2 + offset_12 = r2 + 16 이다.
또 cond field 인 [31:28]이 1110 이므로 (p.112)에 따라 항상 실행하라는 명령어이다.
또 Rd[15:12]는 0000 으로 r0 를 가리킨다. 따라서 이 명령어는 r0 에 있는 값을 r2+16 에 store 하라는 명령어로 STR r0, [r2,#16] 이다.
d) What is the meaning of the instruction? : 메모리에 r0 에 들어있는 값을 r2+16 의 주소를 가진 저장한다.

Address 00B | Instruction E5820014

a) Change to binary format: 1110_0101_1_0_0_0_0010_0000_000000010100
b) Write assembly code: STR r0, [r2, #20]
c) Describe why you wrote the assembly code like above:
A3.1(p.110)에서 [27:25]가 010 이면 load/store immediate 명령어이다. 또 [20]인 L 비트가 0 이므로 store 명령어이다.(p.460).
U 비트 [23]은 1, B 비트[22]는 0 이므로 offset_12[11:0]을 Rn[19:16]인 r2 에 더해 address 를 구한다. address=r2 + offset_12 = r2 + 20 이다.
또 cond field 인 [31:28]이 1110 이므로 (p.112)에 따라 항상 실행하라는 명령어이다.
또 Rd[15:12]는 0000 으로 r0 를 가리킨다. 따라서 이 명령어는 r0 에 있는 값을 r2+20 에 store 하라는 명령어로 STR r0, [r2,#20] 이다.
d) What is the meaning of the instruction? : 메모리에 r0 에 들어있는 값을 r2+20 의 주소를 가진 저장한다.

Address 00C | Instruction E5820018

a) Change to binary format: 1110_ 0101_1_0_0_0_0010_0000_000000011000
b) Write assembly code: STR r0, [r2, #24]
c) Describe why you wrote the assembly code like above:
A3.1(p.110)에서 [27:25]가 010 이면 load/store immediate 명령어이다. 또 [20]인 L 비트가 0 이므로 store 명령어이다.(p.460).
U 비트 [23]은 1, B 비트[22]는 0 이므로 offset_12[11:0]을 Rn[19:16]인 r2 에 더해 address 를 구한다. address=r2 + offset_12 = r2 + 24 이다.
또 cond field 인 [31:28]이 1110 이므로 (p.112)에 따라 항상 실행하라는 명령어이다.
또 Rd[15:12]는 0000 으로 r0 를 가리킨다. 따라서 이 명령어는 r0 에 있는 값을 r2+24 에 store 하라는 명령어로 STR r0, [r2,#24] 이다.
d) What is the meaning of the instruction? : 메모리에 r0 에 들어있는 값을 r2+24 의 주소를 가진 저장한다.

Address 00D | Instruction E582001C

a) Change to binary format: 1110_0101_1_0_0_0_0010_0000_000000011100
b) Write assembly code: STR r0, [r2, #28]
c) Describe why you wrote the assembly code like above:
A3.1(p.110)에서 [27:25]가 010 이면 load/store immediate 명령어이다. 또 [20]인 L 비트가 0 이므로 store 명령어이다.(p.460).
U 비트 [23]은 1, B 비트[22]는 0 이므로 offset_12[11:0]을 Rn[19:16]인 r2 에 더해 address 를 구한다.

address=r2 + offset_12 = r2 + 28 다.
또 cond field 인 [31:28]이 1110 이므로 (p.112)에 따라 항상 실행하라는 명령어이다.
또 Rd[15:12]는 0000 으로 r0 를 가리킨다. 따라서 이 명령어는 r0 에 있는 값을 r2+28 에 store 하라는 명령어로 STR r0, [r2,#28] 이다.

d) What is the meaning of the instruction? : 메모리에 r0 에 들어있는 값을 r+28 의 주소를 가진 저장한다.

Address 00E | Instruction E5820020

a) Change to binary format: 1110/0101/1000/0010/0000/0000/0010/0000
b) Write assembly code: STR r0, [r2, #32]
c) Describe why you wrote the assembly code like above:
A3.1(p.110)에서 [27:25]가 010 이면 load/store immediate 명령어이다. 또 [20]인 L 비트가 0 이므로 store 명령어이다.(p.460).
U 비트 [23]은 1, B 비트[22]는 0 이므로 offset_12[11:0]을 Rn[19:16]인 r2 에 더해 address 를 구한다.
address=r2 + offset_12 = r2 +32 이다.
또 cond field 인 [31:28]이 1110 이므로 (p.112)에 따라 항상 실행하라는 명령어이다.
또 Rd[15:12]는 0000 으로 r0 를 가리킨다. 따라서 이 명령어는 r0 에 있는 값을 r2+32 에 store 하라는 명령어로 STR r0, [r2,#32] 이다.
d) What is the meaning of the instruction? : 메모리에 r0 에 들어있는 값을 r2+32 의 주소를 가진 저장한다.

Address 00F | Instruction E5820024

a) Change to binary format: 1110/0101/1000/0010/0000/0000/0010/0100
b) Write assembly code: STR r0, [r2, #36]
c) Describe why you wrote the assembly code like above:
A3.1(p.110)에서 [27:25]가 010 이면 load/store immediate 명령어이다. 또 [20]인 L 비트가 0 이므로 store 명령어이다.(p.460).
U 비트 [23]은 1, B 비트[22]는 0 이므로 offset_12[11:0]을 Rn[19:16]인 r2 에 더해 address 를 구한다.
address=r2 + offset_12 = r2 + 36 이다.
또 cond field 인 [31:28]이 1110 이므로 (p.112)에 따라 항상 실행하라는 명령어이다.
또 Rd[15:12]는 0000 으로 r0 를 가리킨다. 따라서 이 명령어는 r0 에 있는 값을 r2+36 에 store 하라는 명령어로 STR r0, [r2,#36] 이다.
d) What is the meaning of the instruction? : 메모리에 r0 에 들어있는 값을 r2+36 의 주소를 가진 저장한다.

Address 010 | Instruction E3A0003F

a) Change to binary format: 1110/0011/1010/0000/0000/0000/0011/1111
b) Write assembly code: MOV r0, #63
c) Describe why you wrote the assembly code like above:
A3.1(p.110)에서 [27:25]가 001 이고 [23:23]이 10 이 아니면 data processing immediate 명령어이다.
또 A3.4(p.115)에서 opcode 가 1101 이면 MOV 명령어이므로 이 명령어는 MOV 명령어이다.
또 S 비트[20]은 cpsr 을 업데이트 할지 정하는 비트인데 0 으로 설정되어 있으므로 cpsr 을 업데이트 하지 않는다.(p.218)
또 Rd field[15:12]가 0000 으로 r0 를 가리킨다.
또 cond field 인 [31:28]이 1110 이므로 (p.112)에 따라 항상 실행하라는 명령어이다.
Rotate_imm field[11:8]에 2 를 곱한 만큼 immed_8 field[7:0]을 right rotate 해서 rd 에 들어갈 값을 정하는데 여기선 [11:8]이 0 이므로 10 진수 63 를 나타내는[7:0]이 그대로 들어간다.
따라서 이 명령어는 MOV r0, #63 이다.
d) What is the meaning of the instruction? : r0 에 63 을 저장하는 명령어이다.

Address 011 | Instruction E5820028

a) Change to binary format: 1110/0101/1000/0010/0000/0000/0010/1000
b) Write assembly code: STR r0, [r2, #40]
c) Describe why you wrote the assembly code like above:
A3.1(p.110)에서 [27:25]가 010 이면 load/store immediate 명령어이다. 또 [20]인 L 비트가 0 이므로 store 명령어이다.(p.460).
U 비트 [23]은 1, B 비트[22]는 0 이므로 offset_12[11:0]을 Rn[19:16]인 r2 에 더해 address 를 구한다.
address=r2 + offset_12 = r2 + 40 이다.
또 cond field 인 [31:28]이 1110 이므로 (p.112)에 따라 항상 실행하라는 명령어이다.
또 Rd[15:12]는 0000 으로 r0 를 가리킨다. 따라서 이 명령어는 r0 에 있는 값을 r2+40 에 store 하라는 명령어로 STR r0, [r2,#40] 이다.
d) What is the meaning of the instruction? : 메모리에 r0 에 들어있는 값을 r2+40 의 주소를 가진 저장한다.

Address 012 | Instruction E3A00008

a) Change to binary format: 1110/0011/1010/0000/0000/0000/0000/1000
b) Write assembly code: MOV r0, #8
e) Describe why you wrote the assembly code like above:
A3.1(p.110)에서 [27:25]가 001 이고 [23:23]이 10 이 아니면 data processing immediate 명령어이다.
또 A3.4(p.115)에서 opcode 가 1101 이면 MOV 명령어이므로 이 명령어는 MOV 명령어이다.
또 S 비트[20]은 cpsr 을 업데이트 할지 정하는 비트인데 0 으로 설정되어 있으므로 cpsr 을 업데이트 하지

않는다.(p.218)
또 Rd field[15:12]가 0000 으로 r0 를 가리킨다.
또 cond field 인 [31:28]이 1110 이므로 (p.112)에 따라 항상 실행하라는 명령어이다.
Rotate_imm field[11:8]에 2 를 곱한 만큼 immed_8 field[7:0]을 right rotate 해서 rd 에 들어갈 값을 정하는데 여기선 [11:8]이 0 이므로 10 진수 8 을 나타내는[7:0]이 그대로 들어간다.
따라서 이 명령어는 MOV r0, #8 이다.

c)

d) What is the meaning of the instruction? : r0 에 8 을 저장하는 명령어이다.

Address 013 | Instruction E582002C

a) Change to binary format: 1110/0101/1000/0010/0000/0000/0010/1100
b) Write assembly code: STR r0, [r2, #44]
c) Describe why you wrote the assembly code like above
A3.1(p.110)에서 [27:25]가 010 이면 load/store immediate 명령어이다. 또 [20]인 L 비트가 0 이므로 store 명령어이다.(p.460).
U 비트 [23]은 1, B 비트[22]는 0 이므로 offset_12[11:0]을 Rn[19:16]인 r2 에 더해 address 를 구한다.
address=r2 + offset_12 = r2 + 44 이다.
또 cond field 인 [31:28]이 1110 이므로 (p.112)에 따라 항상 실행하라는 명령어이다.
또 Rd[15:12]는 0000 으로 r0 를 가리킨다. 따라서 이 명령어는 r0 에 있는 값을 r2+44 에 store 하라는 명령어로 STR r0, [r2,#44] 이다
d) What is the meaning of the instruction? : 메모리에 r0 에 들어있는 값을 r2+44 의 주소를 가진 저장한다.

Address 014 | Instruction E59F3E9C

a) Change to binary format: 1110/0101/1001/1111/0011/1110/1001/1100
b) Write assembly code: LDR r3, [pc, #3740]
c) Describe why you wrote the assembly code like above:
A3.1(p.110)에서 [27:25]가 010 이면 load/store immediate 명령어이다. 또 [20]인 L 비트가 1 이므로 load 명령어이다.(p.460).
U 비트 [23]은 1, B 비트[22]는 0 이므로 offset_12[11:0]을 Rn[19:16]인 r15 에 더해 address 를 구한다.
address=r15 + offset_12 = r15 + 3740 이다.
또 cond field 인 [31:28]이 1110 이므로 (p.112)에 따라 항상 실행하라는 명령어이다.
단, 실제 실행시에는 PC 에 8 을 더한 후 3740 를 더해서 address 가 계산된다.
또 Rd[15:12]는 0011 으로 r3 를 가리킨다. 따라서 이 명령어는 r3 에 pc+3740 에 있는 값을 load 하라는 명령어로 LDR r3, [pc, #3740] 이다.

d) What is the meaning of the instruction? : 메모리주소 pc+3740 에 있는 값을 r3 레지스터에 저장하는 명령어이다.

Address 015 | Instruction E59F1E9C

a) Change to binary format: 1110/0101/1001/1111/0001/1110/1001/1100
b) Write assembly code: LDR r1, [pc, #3740]
c) Describe why you wrote the assembly code like above:
A3.1(p.110)에서 [27:25]가 010 이면 load/store immediate 명령어이다. 또 [20]인 L 비트가 1 이므로 load 명령어이다.(p.460).
U 비트 [23]은 1, B 비트[22]는 0 이므로 offset_12[11:0]을 Rn[19:16]인 r15 에 더해 address 를 구한다.
address=r15 + offset_12 = r15 + 3740 이다.
또 cond field 인 [31:28]이 1110 이므로 (p.112)에 따라 항상 실행하라는 명령어이다.
단, 실제 실행시에는 PC 에 8 을 더한 후 3740 를 더해서 address 가 계산된다.
또 Rd[15:12]는 0001 으로 r1 를 가리킨다. 따라서 이 명령어는 r1 에 pc+3740 에 있는 값을 load 하라는 명령어로 LDR r1, [pc, #3740] 이다.
d) What is the meaning of the instruction? : 메모리주소 pc+3740 에 있는 값을 r1 레지스터에 저장하는 명령어이다.

Address 016 | Instruction E5831000

a) Change to binary format: 1110/0101/1000/0011/0001/0000/0000/0000
b) Write assembly code: STR r1, [r3, #0]
c) Describe why you wrote the assembly code like above:
A3.1(p.110)에서 [27:25]가 010 이면 load/store immediate 명령어이다. 또 [20]인 L 비트가 0 이므로 store 명령어이다.(p.460).
U 비트 [23]은 1, B 비트[22]는 0 이므로 offset_12[11:0]을 Rn[19:16]인 r3 에 더해 address 를 구한다.
address=r3 + offset_12 = r3 + 0 이다.
또 cond field 인 [31:28]이 1110 이므로 (p.112)에 따라 항상 실행하라는 명령어이다.
또 Rd[15:12]는 0001 으로 r1 를 가리킨다. 따라서 이 명령어는 r1 에 있는 값을 r3 에 store 하라는 명령어로 STR r1, [r3,#0] 이다
d) What is the meaning of the instruction? : r1 레지스터에 있는 값을 메모리주소 r3+0 에 저장하는 명령어이다.

Address 017 | Instruction E59F9E98

a) Change to binary format: 1110/0101/1001/1111/1001/1110/1001/1000

b) Write assembly code: LDR r9, [pc, #3736]

c) Describe why you wrote the assembly code like above:

A3.1(p.110)에서 [27:25]가 010 이면 load/store immediate 명령어이다. 또 [20]인 L 비트가 1 이므로 load 명령어이다.(p.460).

U 비트 [23]은 1, B 비트[22]는 0 이므로 offset_12[11:0]을 Rn[19:16]인 r15 에 더해 address 를 구한다.

address=r15 + offset_12 = r15 + 3736 이다.

또 cond field 인 [31:28]이 1110 이므로 (p.112)에 따라 항상 실행하라는 명령어이다.

단, 실제 실행시에는 PC 에 8 을 더한 후 3736 를 더해서 address 가 계산된다.

또 Rd[15:12]는 1001 으로 r9 를 가리킨다. 따라서 이 명령어는 r9 에 pc+3736 에 있는 값을 load 하라는 명령어로 LDR r9, [pc, #3740] 이다.

d) What is the meaning of the instruction? : 메모리주소 pc+3736 에 있는 값을 r9 레지스터에 저장하는 명령어이다.

Address 018 | Instruction E3A08000

a) Change to binary format: 1110/001/1101/0/0000/1000/0000/0000/0000

b) Write assembly code: MOV r8, #0

c) Describe why you wrote the assembly code like above:

A3.1(p.110)에서 [27:25]가 001 이고 [23:23]이 10 이 아니면 data processing immediate 명령어이다.

또 A3.4(p.115)에서 opcode 가 1101 이면 MOV 명령어이므로 이 명령어는 MOV 명령어이다.

또 S 비트[20]은 cpsr 을 업데이트 할지 정하는 비트인데 0 으로 설정되어 있으므로 cpsr 을 업데이트 하지 않는다.(p.218)

또 Rd field[15:12]가 1000 으로 r8 를 가리킨다.

또 cond field 인 [31:28]이 1110 이므로 (p.112)에 따라 항상 실행하라는 명령어이다.

Rotate_imm field[11:8]에 2 를 곱한 만큼 immed_8 field[7:0]을 right rotate 해서 rd 에 들어갈 값을 정하는데 여기선 [11:8]이 0 이므로 10 진수 0 을 나타내는[7:0]이 그대로 들어간다.

따라서 이 명령어는 MOV r8, #0 이다.

d) What is the meaning of the instruction? : r8 에 0 을 저장하는 명령어이다.

Address 019 | Instruction E5898000

a) Change to binary format: 1110/0101/1000/1001/1000/0000/0000/0000

b) Write assembly code: STR r8, [r9, #0]

c) Describe why you wrote the assembly code like above:

A3.1(p.110)에서 [27:25]가 010 이면 load/store immediate 명령어이다. 또 [20]인 L 비트가 0 이므로 store 명령어이다.(p.460).

U 비트 [23]은 1, B 비트[22]는 0 이므로 offset_12[11:0]을 Rn[19:16]인 r9 에 더해 address 를 구한다.

address=r15 + offset_12 = r9 + 0 이다.

또 cond field 인 [31:28]이 1110 이므로 (p.112)에 따라 항상 실행하라는 명령어이다.
또 Rd[15:12]는 1000 으로 r8 를 가리킨다. 따라서 이 명령어는 r9+0 에 r8 에 있는 값을 store 하라는 명령어로 STR r8, [r9, #0] 이다.

d) What is the meaning of the instruction? : 메모리주소 r9+0 에 r8 에 있는 값을 저장하는 명령어이다.

Address 01A | Instruction E5898004

a) Change to binary format: 1110/0101/1000/1001/1000/0000/0000/0100
b) Write assembly code: STR r8, [r9, #4]
c) Describe why you wrote the assembly code like above:
A3.1(p.110)에서 [27:25]가 010 이면 load/store immediate 명령어이다. 또 [20]인 L 비트가 0 이므로 store 명령어이다.(p.460).
U 비트 [23]은 1, B 비트[22]는 0 이므로 offset_12[11:0]을 Rn[19:16]인 r9 에 더해 address 를 구한다.
address=r15 + offset_12 = r9 + 4 이다.
또 cond field 인 [31:28]이 1110 이므로 (p.112)에 따라 항상 실행하라는 명령어이다.
또 Rd[15:12]는 1000 으로 r8 를 가리킨다. 따라서 이 명령어는 r9+4 에 r8 에 있는 값을 store 하라는 명령어로 STR r8, [r9, #4] 이다.
d) What is the meaning of the instruction? : 메모리주소 r9+4 에 r8 에 있는 값을 저장하는 명령어이다.

Address 01B | Instruction E5898008

a) Change to binary format: 1110/0101/1000/1001/1000/0000/0000/1000
b) Write assembly code: STR r8, [r9, #8]
c) Describe why you wrote the assembly code like above:
A3.1(p.110)에서 [27:25]가 010 이면 load/store immediate 명령어이다. 또 [20]인 L 비트가 0 이므로 store 명령어이다.(p.460).
U 비트 [23]은 1, B 비트[22]는 0 이므로 offset_12[11:0]을 Rn[19:16]인 r9 에 더해 address 를 구한다.
address=r15 + offset_12 = r9 +8 이다.
또 cond field 인 [31:28]이 1110 이므로 (p.112)에 따라 항상 실행하라는 명령어이다.
또 Rd[15:12]는 1000 으로 r8 를 가리킨다. 따라서 이 명령어는 r9+8 에 r8 에 있는 값을 store 하라는 명령어로 STR r8, [r9, #8] 이다.
d) What is the meaning of the instruction? : 메모리주소 r9+8 에 r8 에 있는 값을 저장하는 명령어이다.

Address 01C | Instruction E589800C

a) Change to binary format: 1110/0101/1000/1001/1000/0000/0000/1100

b) Write assembly code: STR r8, [r9, #12]

c) Describe why you wrote the assembly code like above:

A3.1(p.110)에서 [27:25]가 010 이면 load/store immediate 명령어이다. 또 [20]인 L 비트가 0 이므로 store 명령어이다.(p.460).

U 비트 [23]은 1, B 비트[22]는 0 이므로 offset_12[11:0]을 Rn[19:16]인 r9 에 더해 address 를 구한다.

address=r15 + offset_12 = r9 + 12 이다.

또 cond field 인 [31:28]이 1110 이므로 (p.112)에 따라 항상 실행하라는 명령어이다.

또 Rd[15:12]는 1000 으로 r8 를 가리킨다. 따라서 이 명령어는 r9+12 에 r8 에 있는 값을 store 하라는 명령어로 STR r8, [r9, #12] 이다.

d) What is the meaning of the instruction? : 메모리주소 r9+12 에 r8 에 있는 값을 저장하는 명령어이다.

Address 01D | Instruction E5898010

a) Change to binary format: 1110/0101/1000/1001/1000/0000/0001/0000

b) Write assembly code: STR r8, [r9, #16]

c) Describe why you wrote the assembly code like above:

A3.1(p.110)에서 [27:25]가 010 이면 load/store immediate 명령어이다. 또 [20]인 L 비트가 0 이므로 store 명령어이다.(p.460).

U 비트 [23]은 1, B 비트[22]는 0 이므로 offset_12[11:0]을 Rn[19:16]인 r9 에 더해 address 를 구한다.

address=r15 + offset_12 = r9 + 16 이다.

또 cond field 인 [31:28]이 1110 이므로 (p.112)에 따라 항상 실행하라는 명령어이다.

또 Rd[15:12]는 1000 으로 r8 를 가리킨다. 따라서 이 명령어는 r9+16 에 r8 에 있는 값을 store 하라는 명령어로 STR r8, [r9, #16] 이다.

d) What is the meaning of the instruction? : : 메모리주소 r9+16 에 r8 에 있는 값을 저장하는 명령어이다.

Address 01E | Instruction E5898014

a) Change to binary format: 1110/0101/1000/1001/1000/0000/0001/0100

b) Write assembly code: STR r8, [r9, #20]

c) Describe why you wrote the assembly code like above:

A3.1(p.110)에서 [27:25]가 010 이면 load/store immediate 명령어이다. 또 [20]인 L 비트가 0 이므로 store 명령어이다.(p.460).

U 비트 [23]은 1, B 비트[22]는 0 이므로 offset_12[11:0]을 Rn[19:16]인 r9 에 더해 address 를 구한다.

address=r15 + offset_12 = r9 + 20 이다.

또 cond field 인 [31:28]이 1110 이므로 (p.112)에 따라 항상 실행하라는 명령어이다.

또 Rd[15:12]는 1000 으로 r8 를 가리킨다. 따라서 이 명령어는 r9+20 에 r8 에 있는 값을 store 하라는 명령어로 STR r8, [r9, #20] 이다.

d) What is the meaning of the instruction? : 메모리주소 r9+20 에 r8 에 있는 값을 저장하는 명령어이다.

Address 01F | Instruction E5898018

a) Change to binary format: 1110/0101/1000/1001/1000/0000/0001/1000
b) Write assembly code: STR r8, [r9, #24]
c) Describe why you wrote the assembly code like above:
A3.1(p.110)에서 [27:25]가 010 이면 load/store immediate 명령어이다. 또 [20]인 L 비트가 0 이므로 store 명령어이다.(p.460).
U 비트 [23]은 1, B 비트[22]는 0 이므로 offset_12[11:0]을 Rn[19:16]인 r9 에 더해 address 를 구한다.
address=r15 + offset_12 = r9 + 24 이다.
또 cond field 인 [31:28]이 1110 이므로 (p.112)에 따라 항상 실행하라는 명령어이다.
또 Rd[15:12]는 1000 으로 r8 를 가리킨다. 따라서 이 명령어는 r9+24 에 r8 에 있는 값을 store 하라는 명령어로 STR r8, [r9, #24] 이다
d) What is the meaning of the instruction? : 메모리주소 r9+24 에 r8 에 있는 값을 저장하는 명령어이다.

Address 020 | Instruction E59FDE78

a) Change to binary format: 1110/0101/1001/1111/1101/1110/0111/1000
b) Write assembly code: LDR sp, [pc, #3704]
c) Describe why you wrote the assembly code like above:
A3.1(p.110)에서 [27:25]가 010 이면 load/store immediate 명령어이다. 또 [20]인 L 비트가 1 이므로 load 명령어이다.(p.460).
U 비트 [23]은 1, B 비트[22]는 0 이므로 offset_12[11:0]을 Rn[19:16]인 r15 에 더해 address 를 구한다.
address=r15 + offset_12 = r15 + 3704 이다.
또 cond field 인 [31:28]이 1110 이므로 (p.112)에 따라 항상 실행하라는 명령어이다.
단, 실제 실행시에는 PC 에 8 을 더한 후 3784 를 더해서 address 가 계산된다.
또 Rd[15:12]는 1101 으로 sp 를 가리킨다. 따라서 이 명령어는 sp 에 pc+3704 에 있는 값을 load 하라는 명령어로 LDR sp, [pc, #3704] 이다.
d) What is the meaning of the instruction? : sp 에 메모리주소 pc+3704 에 있는 값을 저장하라는 명령어이다.

Address 021 | Instruction E5931200

a) Change to binary format: 1110/0101/1001/0011/0001/0010/0000/0000
b) Write assembly code: LDR r1, [r3, #512]
c) Describe why you wrote the assembly code like above:
A3.1(p.110)에서 [27:25]가 010 이면 load/store immediate 명령어이다. 또 [20]인 L 비트가 1 이므로 load 명령어이다.(p.460).
U 비트 [23]은 1, B 비트[22]는 0 이므로 offset_12[11:0]을 Rn[19:16]인 r3 에 더해 address 를 구한다.
address=r15 + offset_12 = r3 + 512 이다.
또 cond field 인 [31:28]이 1110 이므로 (p.112)에 따라 항상 실행하라는 명령어이다.
단, 실제 실행시에는 PC 에 8 을 더한 후 512 를 더해서 address 가 계산된다.
또 Rd[15:12]는 0001 으로 r1 를 가리킨다. 따라서 이 명령어는 r1 에 rc+512 에 있는 값을 load 하라는 명령어로 LDR r1, [r3, #512] 이다.
d) What is the meaning of the instruction? : r1 에 메모리주소 r3+512 에 있는 값을 저장하라는 명령어이다.

Address 022 | Instruction E3510001

a) Change to binary format: 1110/001/1010/1/0001/0000/0000/0000/0001
b) Write assembly code: CMP r1, #1
c) Describe why you wrote the assembly code like above:
A3.1(p.110)에서 [27:25]가 001 이고 [23:23]이 10 이 아니면 data processing immediate 명령어이다.
또 A3.4(p.115)에서 opcode 가 11010 이면 CMP 명령어이므로 이 명령어는 CMP 명령어이다.
또 Rn field[19:16]가 0001 으로 r1 를 가리킨다.
또 cond field 인 [31:28]이 1110 이므로 (p.112)에 따라 항상 실행하라는 명령어이다.
Rotate_imm field[11:8]에 2 를 곱한 만큼 immed_8 field[7:0]을 right rotate 해서 rd 에 들어갈 값을 정하는데 여기선 [11:8]이 0 이므로 10 진수 1 을 나타내는[7:0]이 그대로 들어간다.
따라서 이 명령어는 CMP r1, #1 이다.
d) What is the meaning of the instruction? : r1 과 1 이 같으면 Z flag 를 set 하라는 명령어이다.

Address 023 | Instruction 0A000000

a) Change to binary format: 0000/1010/0000/0000/0000/0000/0000/0000
b) Write assembly code: BEQ #37
c) Describe why you wrote the assembly code like above:
A3.1(p.110)에서 [27:25]가 101 이면 branch 명령어이다. 따라서 이 명령어는 branch 명령어임을 알 수 있다.
또 [24]인 L 비트가 0 이므로 돌아올 곳을 R14 에 저장하는 BL 명령어가 아닌 B 명령어 임을 알 수 있다.(p.160)
또 cond 비트인 [31:28]이 0000 이므로 (p.112)에 따라 Z flag 가 1 이면 분기하라는 명령어이다. 따라서 BEQ 명령어 임을 알 수 있다.

또 하위 24 비트[23:0]은 offset 으로 다음 과정을 거쳐 pc 에 더해진다.

1. 30bit 로 signed extension 한다. -> msb 가 0 이므로 0 을 채운다.

->00/0000/0000/0000/0000/0000/0000/0000

2. 2bit left shift 로 singed 32bit 로 만든다.-> 0000/0000/0000/0000/0000/0000/0000/0000

3. 이 값을 현재 명령어에 8 이 더해진 값을 가진 pc 에 더한다.

이 32bit 값은 10 진수로 변환하면 0 이다.

따라서 이 명령어의 주소는 (35*4)=140 이므로 현재 pc 값은 140+8 인 148 이다. 따라서 target_address 는 148+0=148 이고 이를 4 로 나누면 37 이다. 따라서 이 명령어는 B #37 이다

d) What is the meaning of the instruction? : z flag 가 set 되어있으면 37 번째 명령어(address 025)로 분기하라는 명령어이다.

Address 024 | Instruction EAFFFFFB

a) Change to binary format: 1110/1010/1111/1111/1111/1111/1111/1011
b) Write assembly code: B #33
c) Describe why you wrote the assembly code like above:

A3.1(p.110)에서 [27:25]가 101 이면 branch 명령어이다. 따라서 이 명령어는 branch 명령어임을 알 수 있다.

또 [24]인 L 비트가 0 이므로 돌아올 곳을 R14 에 저장하는 BL 명령어가 아닌 B 명령어 임을 알 수 있다.(p.160)

또 cond 비트인 [31:28]이 1110 이므로 (p.112)에 따라 항상 분기하라는 명령어이다. 따라서 BAL 명령어 임을 알 수 있다.

또 하위 24 비트[23:0]은 offset 으로 다음 과정을 거쳐 pc 에 더해진다.

1. 30bit 로 signed extension 한다. -> msb 가 1 이므로 1 을 채운다.

->11/1111/1111/1111/1111/1111/1111/1011

2. 2bit left shift 로 singed 32bit 로 만든다.-> 1111/1111/1111/1111/1111/1111/1110/1100

3. 이 값을 현재 명령어에 8 이 더해진 값을 가진 pc 에 더한다.

이 32bit 값은 10 진수로 변환하면 -20 이다.

따라서 이 명령어의 주소는 (36*4)=144 이므로 현재 pc 값은 144+8 인 152 이다. 따라서 target_address 는 152-20=132 이고 이를 4 로 나누면 33 이다. 따라서 이 명령어는 B #33 이다

d) What is the meaning of the instruction? : 33 번째 명령어(address021)로 분기하라는 명령어이다.

Explain the actual execution flow of the instructions(Address 000~024)

000| b#8: pc 가 8 로 설정되어 8 번째 명령어로 분기함

008|ldr r2, [pc, #3784]: pc 에 3784 를 더한 주소로 접근해 해당 값을 r2 에 저장함. 단, pc 는 현재 명령어의 위치 008 에 8 을 더한 0x10 을 나타냄.

009| mov r0, #64: r0 에 64 를 저장.

00A|str r0, [r2, #16]: mem[r2+16]에 r0(64)를 저장.

00B|str r0, [r2, #20]: mem[r2+20]에 r0(64)를 저장.

00C|str r0, [r2, #24] : mem[r2+24]에 r0(64)를 저장.

00D|str r0, [r2, #28] : mem[r2+28]에 r0(64)를 저장.

00E|str r0, [r2, #32] : mem[r2+32]에 r0(64)를 저장.

00F|str r0, [r2, #36] : mem[r2+36]에 r0(64)를 저장.

010|mov r0, #63 : r0 에 63 을 저장

011|str r0, [r2,#40] : mem[r2+40]에 r0(64)를 저장.

012|mov r0, #8 : r0 에 8 을 저장

013|str r0, [r2, #44] : mem[r2+44]에 r0(8)을 저장

014|ldr r3, [pc, #3740] : r3 에 mem[pc+3740]을 저장. 단 pc 는 현재 명령어의 주소에서 8 을 더한 값을 나타냄.

015|ldr r1, [pc, #3740] : r1 에 mem[pc+3740]을 저장. 단 pc 는 현재 명령어의 주소에서 8 을 더한 값을 나타냄.

016|str r1, [r3, #0] : mem[r3+0]에 r1(unknown)을 저장

017|str r9, [pc, #3736] : mem[pc+3736]에 r9 을 저장. 단 pc 는 현재 명령어의 주소에서 8 을 더한 값을 나타냄.

018|mov r8, #0 : r8 에 0 을 저장

019|str r8, [r9, #0] : mem[r9+0]에 r8(0)을 저장

01A|str r8, [r9, #4] : mem[r9+4]에 r8(0)을 저장

01B|str r8, [r9, #8] : mem[r9+8]에 r8(0)을 저장

01C|str r8, [r9, #12] : mem[r9+12]에 r8(0)을 저장

01D|str r8, [r9, #16] : mem[r9+16]에 r8(0)을 저장

01E|str r8, [r9, #20] : mem[r9+20]에 r8(0)을 저장

01F|str r8, [r9, #24] : mem[r9+24]에 r8(0)을 저장

020:str sp, [pc, #3704]: mem[pc+3704]에 sp(r13)(unknown)을 저장

021|str r1, [r3, #512] : mem[r3+512]에 r1(unknown)을 저장

022|cmp r1, #1 : r1(unknown)과 상수 1 이 같으면 z-flag=1 else z-flag=0

023|beq #37 : z-flag=1 이면 37(address025)번째 명령어로 분기 else pc=pc+4

<case1>(r1=1)

pc 가 0x25 가 되어 더 이상 읽을 명령어가 없어 pc 는 계속 증가하나 명령어를 실행하진 못한다.

<case2>(r1!=1)

024| b#33: 33 번째 명령어(address021)로 분기

021| str r1, [r3, #512] : mem[r3+512]에 r1(not 1)을 저장

022|cmp r1, #1 : r1(not 1)과 상수 1 이 다르므로 z-flag=0

023|beq #37 : z-flag=0 이므로 pc=pc+4

(이하 무한루프)

Specify where the execution ends (If not, specify the range repeated in detail)

case1 에서는 별다른 연산없이 pc 만 계속 증가한다.

case2 에서는 address021~address024 를 무한루프를 돈다.