Computer Architecture Assignment #1

Your Student Number: 2022320033

Name: 박종혁

<<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 #008;
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 #001
c) Describe why you wrote the assembly code like above:
    a. Figure A3-1 에 따르면, [25:27] 비트가 ‘101’ 이므로 branch instruction 이다.
    b. Cond field 의 값이 1110 (AL) 이므로 조건 없이 실행된다.
    c. Branch instruction 의 syntax 는 B{L}{cond} <target address> 이다.
    d. A4.1.5 에 따르면, L 비트가 0 이므로 B instruction 이다.
    e. A4.1.5 에 따르면, <target address> 는 다음과 같이 계산된다.
        i. 24-bit immediate 1111 1111 1111 1111 1111 1110 을 30-bit 로 sign-extending 한 결과 11 1111 1111 1111 1111 1111 1111 1110 을 얻는다.
        ii. 얻은 결과를 2-bit shift left 하면
        1111 1111 1111 1111 1111 1111 1111 1000 = -8 (base 10)
        iii. PC 값에 이 값을 더해 target address 를 얻는다.
        {(1*4 + 8) + (-8)} / 4 = 1 (base 10) = 0x001 (base 16)
        iv. 따라서 assembly code 는 B #001 이다.
d) What is the meaning of the instruction? : address 001 로 branch 한다. 자기 자신의 주소이므로 무한 루프에 빠진다.

Address 002 | Instruction EA0000A7

a) Change to binary format: 1110 1010 0000 0000 0000 0000 1010 0111
b) Write assembly code: B #0AB
c) Describe why you wrote the assembly code like above:
    a. Figure A3-1 에 따르면, [25:27] 비트가 ‘101’이므로 branch instruction 이다.
    b. Cond field 의 값이 1110 (AL) 이므로 조건 없이 실행된다.
    c. Branch instruction 의 syntax 는 B{L}{cond} <target address> 이다.
    d. A4.1.5 에 따르면, L 비트가 0 이므로 B instruction 이다.
    e. A4.1.5 에 따르면, <target address> 는 다음과 같이 계산된다.
        i. 24-bit immediate 0000 0000 0000 0000 1010 0111 을 30-bit 로 sign-extending 한 결과 00 0000 0000 0000 0000 0000 1010 0111 을 얻는다.
        ii. 얻은 결과를 2-bit shift left 하면
        0000 0000 0000 0000 0000 0010 1001 1100= 668 (base 10)

iii. PC 값에 이 값을 더해 target address 를 얻는다.
{(2*4 + 8) + (668)} / 4 = 171 (base 10) = 0x0AB (base 16)
iv. 따라서 assembly code 는 B #0AB 이다.

d) What is the meaning of the instruction? : address 0AB 로 branch 한다.

Address 003~005 | Instruction EAFFFFFE

a) Change to binary format: 1110 1010 1111 1111 1111 1111 1111 1110
b) Write assembly code: B #003, B#004, B#005
c) Describe why you wrote the assembly code like above:
   a. Figure A3-1 에 따르면, [25:27] 비트가 '101' 이므로 branch instruction 이다.
   b. Cond field 의 값이 1110 (AL) 이므로 조건 없이 실행된다.
   c. Branch instruction 의 syntax 는 B{L}{cond} <target address> 이다.
   d. A4.1.5 에 따르면, L 비트가 0 이므로 B instruction 이다.
   e. A4.1.5 에 따르면, <target address> 는 다음과 같이 계산된다.
      i. 24-bit immediate 1111 1111 1111 1111 1111 1110 을 30-bit 로 sign-extending 한 결과 11 1111 1111 1111 1111 1111 1111 1110 을 얻는다.
      ii. 얻은 결과를 2-bit shift left 하면
      1111 1111 1111 1111 1111 1111 1111 1000 = -8 (base 10)
      iii. PC 값에 이 값을 더해 target address 를 얻는다.
      {((3 or 4 or 5)*4 + 8) + (-8)} / 4 = 0x003 or 0x004 or 0x005 (base 16)
      iv. 따라서 assembly code 는 B #003, B#004, B#005 이다.

What is the meaning of the instruction? : 각각 address 003, 004, 005 로 branch 한다. 자기 자신의 주소이므로 무한루프에 빠진다.

Address 006 | Instruction EA0000A4

a) Change to binary format: 1110 1010 0000 0000 0000 0000 1010 0100
b) Write assembly code: B #0AC
c) Describe why you wrote the assembly code like above:
   f. Figure A3-1 에 따르면, [25:27] 비트가 '101' 이므로 branch instruction 이다.
   g. Cond field 의 값이 1110 (AL) 이므로 조건 없이 실행된다.
   h. Branch instruction 의 syntax 는 B{L}{cond} <target address> 이다.
   i. A4.1.5 에 따르면, L 비트가 0 이므로 B instruction 이다.
   j. A4.1.5 에 따르면, <target address> 는 다음과 같이 계산된다.

i. 24-bit immediate 0000 0000 0000 0000 1010 0100 을 30-bit 로 sign-extending 한 결과 00 0000 0000 0000 0000 0000 1010 0100 을 얻는다.
ii. 얻은 결과를 2-bit shift left 하면
0000 0000 0000 0000 0000 0010 1001 0000 = 656(base 10)
iii. PC 값에 이 값을 더해 target address 를 얻는다.
{(6 * 4 + 8) + (656)} / 4 = 0x0AC (base 16)
iv. 따라서 assembly code 는 B #0AC 이다.

d) What is the meaning of the instruction? : address 0AC 로 branch 한다.

Address 007 | Instruction EAFFFFFE

a) Change to binary format: 1110 1010 1111 1111 1111 1111 1111 1110
b) Write assembly code: B #007
c) Describe why you wrote the assembly code like above:
   a. Figure A3-1 에 따르면, [25:27] 비트가 '101' 이므로 branch instruction 이다.
   b. Cond field 의 값이 1110 (AL) 이므로 조건 없이 실행된다.
   c. Branch instruction 의 syntax 는 B{L}{cond} <target address> 이다.
   d. A4.1.5 에 따르면, L 비트가 0 이므로 B instruction 이다.
   e. A4.1.5 에 따르면, <target address> 는 다음과 같이 계산된다.
      i. 24-bit immediate 1111 1111 1111 1111 1111 1110 을 30-bit 로 sign-extending 한 결과 11 1111 1111 1111 1111 1111 1111 1110 을 얻는다.
      ii. 얻은 결과를 2-bit shift left 하면
      1111 1111 1111 1111 1111 1111 1111 1000 = -8 (base 10)
      iii. PC 값에 이 값을 더해 target address 를 얻는다.
      {((7)*4 + 8) + (-8)} / 4 = 0x007 (base 16)
      iv. 따라서 assembly code 는 B #007 이다.

What is the meaning of the instruction? : address 007 로 branch 한다. 자기 자신의 주소이므로 무한 루프에 빠진다.

Address 008 | Instruction E59F2EC8

a) Change to binary format: 1110 0101 1001 1111 0010 1110 1100 1000
b) Write assembly code: LDR r2, [pc, #+0xEC8]
c) Describe why you wrote the assembly code like above:

a. Figure A3-1 에 따르면, [25:27] 비트가 '010' 이므로 Load/store immediate offset instruction 이다.
b. Cond field 의 값이 1110 (AL) 이므로 조건 없이 실행된다.
c. Figure 4-14 에 따르면, L 비트가 '1' 이므로 Load instruction 이다.
d. Load instuction 의 Syntax 는 LDR {<cond>} <Rd> <Addressing_mode> 이다.
e. A 5.2.1 에 따르면, P == 1, W == 0 이므로 base register 의 값은 업데이트되지 않는다.
f. Rn == 1111 (r15), Rd == 0010 (r2), offset_12 == 1110 1100 1000 (0xEC8)이다.
g. A5.2.2 에 따르면, U == 1 이므로 address 는 Rn 레지스터에 들어있는 주소값에 offset_12 를 더한 값이다.
h. 즉, r2 레지스터에 (r15 레지스터의 값 + offset_12) 주소에 있는 값을 저장하는 연산이다.
i. 그런데 r15 레지스터는 pc 값을 저장하는 레지스터이므로 address mode 는 [pc, #0xEC8] 이다.

d) What is the meaning of the instruction? : r2 레지스터에 메모리의 (pc + 0xEC8) 주소에 들어있는 값을 저장한다.

Address 009 | Instruction E3A00040

a) Change to binary format: 1110 0011 1010 0000 0000 0000 0100 0000
b) Write assembly code: MOV r0, #0x40
c) Describe why you wrote the assembly code like above:
   a. Figure A3-1 에 따르면, [25:27]비트가 '001'이고, [23:24]비트가 '10'이 아니므로 Data processing instruction 이다. 그중에서도 opcode 가 '1101'이므로 MOV instruction 이다.
   b. MOV instruction 의 syntax 는 MOV {<cond>}{S} <Rd>, <shifter_operand> 이다.
   c. Cond field 의 값이 '1110'이므로 조건 없이 실행된다.
   d. A4.1.35 에 따르면, I == 1 이므로 Immediate value 를 사용한다.
   e. A4.1.35 에 따르면, S==0 이므로 MOV 명령어이고, CPSR 을 업데이트하지 않는다.
   f. Rd == 0000 (r0)
   g. A5.1.3 에 따르면, rotate_imm == 0000 이고 immed_8 은 0100 0000 (0x40) 이므로 <immediate> = 0x40, shifter_carry_out = C flag 이다.
   h. 따라서 r0 레지스터에 상수 0x40 을 복사하는 연산이다.
d) What is the meaning of the instruction? : r0 레지스터에 상수 0x40 을 복사한다.

Address 00A | Instruction E5820010

a) Change to binary format: 1110 0101 1000 0010 0000 0000 0001 0000
b) Write assembly code: STR r0, [r2, #+0x010]
c) Describe why you wrote the assembly code like above:
    a. Figure A3-1 에 따르면, [25:27] 비트가 '010' 이므로 Load/store immediate offset instruction 이다.
    b. Cond field 의 값이 1110 (AL) 이므로 조건 없이 실행된다.
    c. Figure 4-14 에 따르면, L 비트가 '0' 이므로 Store instruction 이다.
    d. Store instuction 의 Syntax 는 STR {<cond>} <Rd> <Addressing_mode> 이다.
    e. A 5.2.1 에 따르면, P == 1, W == 0 이므로 base register 의 값은 업데이트되지 않는다.
    f. Rn == 0010 (r2), Rd == 0000 (r0), offset_12 == 0000 0001 0000 (0x010)이다.
    g. A5.2.2 에 따르면, U == 1 이므로 address 는 Rn 레지스터에 들어있는 주소값에 offset_12 를 더한 값이다.
    h. 즉, 메모리의 (r2 + 0x010) 번째 주소에 r0 레지스터의 값을 저장하는 연산이다.
d) What is the meaning of the instruction? : r0 레지스터의 값을 메모리의 (r2 + 0x010) 주소에 저장한다.

Address 00B | Instruction E5820014

a) Change to binary format: 1110 0101 1000 0010 0000 0000 0001 0100
b) Write assembly code: STR r0, [r2, #+0x014]
c) Describe why you wrote the assembly code like above:
    a. Figure A3-1 에 따르면, [25:27] 비트가 '010' 이므로 Load/store immediate offset instruction 이다.
    b. Cond field 의 값이 1110 (AL) 이므로 조건 없이 실행된다.
    c. Figure 4-14 에 따르면, L 비트가 '0' 이므로 Store instruction 이다.
    d. Store instuction 의 Syntax 는 STR {<cond>} <Rd> <Addressing_mode> 이다.
    e. A 5.2.1 에 따르면, P == 1, W == 0 이므로 base register 의 값은 업데이트되지 않는다.
    f. Rn == 0010 (r2), Rd == 0000 (r0), offset_12 == 0000 0001 0100 (0x014)이다.
    g. A5.2.2 에 따르면, U == 1 이므로 address 는 Rn 레지스터에 들어있는 주소값에 offset_12 를 더한 값이다.
    h. 즉, 메모리의 (r2 + 0x014) 번째 주소에 r0 레지스터의 값을 저장하는 연산이다.

d) What is the meaning of the instruction? : r0 레지스터의 값을 메모리의 (r2 + 0x014) 주소에 저장한다.

Address 00C | Instruction E5820018

a) Change to binary format: 1110 0101 1000 0010 0000 0000 0001 1000
b) Write assembly code: STR r0, [r2, #+0x018]
c) Describe why you wrote the assembly code like above:
    a. Figure A3-1 에 따르면, [25:27] 비트가 '010' 이므로 Load/store immediate offset instruction 이다.
    b. Cond field 의 값이 1110 (AL) 이므로 조건 없이 실행된다.
    c. Figure 4-14 에 따르면, L 비트가 '0' 이므로 Store instruction 이다.
    d. Store instuction 의 Syntax 는 STR {<cond>} <Rd> <Addressing_mode> 이다.
    e. A 5.2.1 에 따르면, P == 1, W == 0 이므로 base register 의 값은 업데이트되지 않는다.
    f. Rn == 0010 (r2), Rd == 0000 (r0), offset_12 == 0000 0001 1000 (0x018)이다.
    g. A5.2.2 에 따르면, U == 1 이므로 address 는 Rn 레지스터에 들어있는 주소값에 offset_12 를 더한 값이다.
    h. 즉, 메모리의 (r2 + 0x018) 번째 주소에 r0 레지스터의 값을 저장하는 연산이다.
d) What is the meaning of the instruction? : r0 레지스터의 값을 메모리의 (r2 + 0x018) 주소에 저장한다.

Address 00D | Instruction E582001C

a) Change to binary format: 1110 0101 1000 0010 0000 0000 0001 1100
b) Write assembly code: STR r0, [r2, #+0x01C]
c) Describe why you wrote the assembly code like above:
    a. Figure A3-1 에 따르면, [25:27] 비트가 '010' 이므로 Load/store immediate offset instruction 이다.
    b. Cond field 의 값이 1110 (AL) 이므로 조건 없이 실행된다.
    c. Figure 4-14 에 따르면, L 비트가 '0' 이므로 Store instruction 이다.
    d. Store instuction 의 Syntax 는 STR {<cond>} <Rd> <Addressing_mode> 이다.
    e. A 5.2.1 에 따르면, P == 1, W == 0 이므로 base register 의 값은 업데이트되지 않는다.
    f. Rn == 0010 (r2), Rd == 0000 (r0), offset_12 == 0000 0001 1100 (0x01C)이다.

g. A5.2.2 에 따르면, U == 1 이므로 address 는 Rn 레지스터에 들어있는 주소값에 offset_12 를 더한 값이다.
h. 즉, 메모리의 (r2 + 0x01C) 번째 주소에 r0 레지스터의 값을 저장하는 연산이다.

d) What is the meaning of the instruction? : r0 레지스터의 값을 메모리의 (r2 + 0x01C) 주소에 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, #+0x020]
c) Describe why you wrote the assembly code like above:
   a. Figure A3-1 에 따르면, [25:27] 비트가 '010' 이므로 Load/store immediate offset instruction 이다.
   b. Cond field 의 값이 1110 (AL) 이므로 조건 없이 실행된다.
   c. Figure 4-14 에 따르면, L 비트가 '0' 이므로 Store instruction 이다.
   d. Store instuction 의 Syntax 는 STR {<cond>} <Rd> <Addressing_mode> 이다.
   e. A 5.2.1 에 따르면, P == 1, W == 0 이므로 base register 의 값은 업데이트되지 않는다.
   f. Rn == 0010 (r2), Rd == 0000 (r0), offset_12 == 0000 0010 0000 (0x020)이다.
   g. A5.2.2 에 따르면, U == 1 이므로 address 는 Rn 레지스터에 들어있는 주소값에 offset_12 를 더한 값이다.
   h. 즉, 메모리의 (r2 + 0x020) 번째 주소에 r0 레지스터의 값을 저장하는 연산이다.
d) What is the meaning of the instruction? : r0 레지스터의 값을 메모리의 (r2 + 0x020) 주소에 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, #+0x024]
c) Describe why you wrote the assembly code like above:
   a. Figure A3-1 에 따르면, [25:27] 비트가 '010' 이므로 Load/store immediate offset instruction 이다.
   b. Cond field 의 값이 1110 (AL) 이므로 조건 없이 실행된다.
   c. Figure 4-14 에 따르면, L 비트가 '0' 이므로 Store instruction 이다.
   d. Store instuction 의 Syntax 는 STR {<cond>} <Rd> <Addressing_mode> 이다.

e. A 5.2.1 에 따르면, P == 1, W == 0 이므로 base register 의 값은 업데이트되지 않는다.
f. Rn == 0010 (r2), Rd == 0000 (r0), offset_12 == 0000 0010 0100 (0x024)이다.
g. A5.2.2 에 따르면, U == 1 이므로 address 는 Rn 레지스터에 들어있는 주소값에 offset_12 를 더한 값이다.
h. 즉, 메모리의 (r2 + 0x024) 번째 주소에 r0 레지스터의 값을 저장하는 연산이다.

d) What is the meaning of the instruction? : r0 레지스터의 값을 메모리의 (r2 + 0x024) 주소에 저장한다.

Address 010 | Instruction E3A0003F

a) Change to binary format: 1110 0011 1010 0000 0000 0000 0011 1111
b) Write assembly code: MOV r0, #0x3F
c) Describe why you wrote the assembly code like above:
a. Figure A3-1 에 따르면, [25:27]비트가 '001'이고, [23:24]비트가 '10'이 아니므로 Data processing instruction 이다. 그중에서도 opcode 가 '1101'이므로 MOV instruction 이다.
b. MOV instruction 의 syntax 는 MOV {<cond>}{S} <Rd>, <shifter_operand> 이다.
c. Cond field 의 값이 '1110' (AL)이므로 조건 없이 실행된다.
d. A4.1.35 에 따르면, I == 1 이므로 Immediate value 를 사용한다.
e. A4.1.35 에 따르면, S==0 이므로 MOV 명령어이고, CPSR 을 업데이트하지 않는다.
f. Rd == 0000 (r0)
g. A5.1.3 에 따르면, rotate_imm == 0000 이고 immed_8 은 0011 1111 (0x3F)이므로 <immediate> = 0x3F, shifter_carry_out = C flag 이다.
h. 따라서 r0 레지스터에 상수 0x3F 을 복사하는 연산이다.

d) What is the meaning of the instruction? : r0 레지스터에 상수 0x3F 을 복사한다.

Address 011 | Instruction E5820028

a) Change to binary format: 1110 0101 1000 0010 0000 0000 0010 1000
b) Write assembly code: STR r0, [r2, #+0x028]
c) Describe why you wrote the assembly code like above:
a. Figure A3-1 에 따르면, [25:27] 비트가 '010' 이므로 Load/store immediate offset instruction 이다.
b. Cond field 의 값이 1110 (AL) 이므로 조건 없이 실행된다.

c. Figure 4-14 에 따르면, L 비트가 '0' 이므로 Store instruction 이다.

d. Store instuction 의 Syntax 는 STR {<cond>} <Rd> <Addressing_mode> 이다.

e. A 5.2.1 에 따르면, P == 1, W == 0 이므로 base register 의 값은 업데이트되지 않는다.

f. Rn == 0010 (r2), Rd == 0000 (r0), offset_12 == 0000 0010 1000 (0x028)이다.

g. A5.2.2 에 따르면, U == 1 이므로 address 는 Rn 레지스터에 들어있는 주소값에 offset_12 를 더한 값이다.

h. 즉, 메모리의 (r2 + 0x028) 번째 주소에 r0 레지스터의 값을 저장하는 연산이다.

d) What is the meaning of the instruction? : r0 레지스터의 값을 메모리의 (r2 + 0x028) 주소에 저장한다.

Address 012 | Instruction E3A00008

a) Change to binary format: 1110 0011 1010 0000 0000 0000 0000 1000
b) Write assembly code: MOV r0, #0x08
c) Describe why you wrote the assembly code like above:

a. Figure A3-1 에 따르면, [25:27]비트가 '001'이고, [23:24]비트가 '10'이 아니므로 Data processing instruction 이다. 그중에서도 opcode 가 '1101'이므로 MOV instruction 이다.

b. MOV instruction 의 syntax 는 MOV {<cond>}{S} <Rd>, <shifter_operand> 이다.

c. Cond field 의 값이 '1110'이므로 조건 없이 실행된다.

d. A4.1.35 에 따르면, I == 1 이므로 Immediate value 를 사용한다.

e. A4.1.35 에 따르면, S==0 이므로 MOV 명령어이고, CPSR 을 업데이트하지 않는다.

f. Rd == 0000 (r0)

g. A5.1.3 에 따르면, rotate_imm == 0000 이고 immed_8 은 0000 1000 (0x08) 이므로 <immediate> = 0x08, shifter_carry_out = C flag 이다.

h. 따라서 r0 레지스터에 상수 0x08 을 복사하는 연산이다.

d) What is the meaning of the instruction? : r0 레지스터에 상수 0x08 을 복사한다.

Address 013 | Instruction E582002C

a) Change to binary format: 1110 0101 1000 0010 0000 0000 0010 1100
b) Write assembly code: STR r0, [r2, #+0x02C]
c) Describe why you wrote the assembly code like above:

a. Figure A3-1 에 따르면, [25:27] 비트가 '010' 이므로 Load/store immediate offset instruction 이다.
b. Cond field 의 값이 1110 (AL) 이므로 조건 없이 실행된다.
c. Figure 4-14 에 따르면, L 비트가 '0' 이므로 Store instruction 이다.
d. Store instuction 의 Syntax 는 STR {<cond>} <Rd> <Addressing_mode> 이다.
e. A 5.2.1 에 따르면, P == 1, W == 0 이므로 base register 의 값은 업데이트되지 않는다.
f. Rn == 0010 (r2), Rd == 0000 (r0), offset_12 == 0000 0010 1100 (0x02C) 이다.
g. A5.2.2 에 따르면, U == 1 이므로 address 는 Rn 레지스터에 들어있는 주소값에 offset_12 를 더한 값이다.
h. 즉, 메모리의 (r2 + 0x02C) 번째 주소에 r0 레지스터의 값을 저장하는 연산이다.

d) What is the meaning of the instruction? : r0 레지스터의 값을 메모리의 (r2 + 0x02C) 주소에 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, #+0xE9C]
c) Describe why you wrote the assembly code like above:
   a. Figure A3-1 에 따르면, [25:27] 비트가 '010' 이므로 Load/store immediate offset instruction 이다.
   b. Cond field 의 값이 1110 (AL) 이므로 조건 없이 실행된다.
   c. Figure 4-14 에 따르면, L 비트가 '1' 이므로 Load instruction 이다.
   d. Load instuction 의 Syntax 는 LDR {<cond>} <Rd> <Addressing_mode> 이다.
   e. A 5.2.1 에 따르면, P == 1, W == 0 이므로 base register 의 값은 업데이트되지 않는다.
   f. Rn == 1111 (r15), Rd == 0011 (r3), offset_12 == 1110 1001 1100 (0xE9C) 이다.
   g. A5.2.2 에 따르면, U == 1 이므로 address 는 Rn 레지스터에 들어있는 주소값에 offset_12 를 더한 값이다.
   h. 즉, 메모리의 (r15 + 0xE9C) 주소에 있는 값을 r3 레지스터에 저장하는 연산이다.
   i. 그런데 r15 레지스터는 pc 값을 저장하므로 pc + 0xE9C 이다.
d) What is the meaning of the instruction? : 메모리의 (pc + 0xE9C) 주소에 있는 값을 r3 레지스터에 저장한다.

Address 015 | Instruction E59F1E9C

a) Change to binary format: 1110 0101 1001 1111 0001 1110 1001 1100
b) Write assembly code: LDR r1, [pc, #+0xE9C]
c) Describe why you wrote the assembly code like above:
    a. Figure A3-1 에 따르면, [25:27] 비트가 '010' 이므로 Load/store immediate offset instruction 이다.
    b. Cond field 의 값이 1110 (AL) 이므로 조건 없이 실행된다.
    c. Figure 4-14 에 따르면, L 비트가 '1' 이므로 Load instruction 이다.
    d. Load instuction 의 Syntax 는 LDR {<cond>} <Rd> <Addressing_mode> 이다.
    e. A 5.2.1 에 따르면, P == 1, W == 0 이므로 base register 의 값은 업데이트되지 않는다.
    f. Rn == 1111 (r15), Rd == 0001 (r1), offset_12 == 1110 1001 1100 (0xE9C)이다.
    g. A5.2.2 에 따르면, U == 1 이므로 address 는 Rn 레지스터에 들어있는 주소값에 offset_12 를 더한 값이다.
    h. 즉, 메모리의 (r15 + 0xE9C) 주소에 있는 값을 r1 레지스터에 저장하는 연산이다.
    i. 그런데 r15 레지스터는 pc 값을 저장하므로 pc + 0xE9C 이다.
d) What is the meaning of the instruction? : 메모리의 (pc + 0xE9C) 주소에 있는 값을 r1 레지스터에 저장한다.

Address 016 | Instruction E5831000

a) Change to binary format: 1110 0101 1000 0011 0001 0000 0000 0000
b) Write assembly code: STR r1, [r3, #+0x000]
c) Describe why you wrote the assembly code like above:
    a. Figure A3-1 에 따르면, [25:27] 비트가 '010' 이므로 Load/store immediate offset instruction 이다.
    b. Cond field 의 값이 1110 (AL) 이므로 조건 없이 실행된다.
    c. Figure 4-14 에 따르면, L 비트가 '0' 이므로 Store instruction 이다.
    d. Store instuction 의 Syntax 는 STR {<cond>} <Rd> <Addressing_mode> 이다.
    e. A 5.2.1 에 따르면, P == 1, W == 0 이므로 base register 의 값은 업데이트되지 않는다.
    f. Rn == 0011 (r3), Rd == 0001 (r1), offset_12 == 0000 0000 0000 (0x000)이다.
    g. A5.2.2 에 따르면, U == 1 이므로 address 는 Rn 레지스터에 들어있는 주소값에 offset_12 를 더한 값이다.
    h. 즉, 메모리의 (r3 + 0x000) 번째 주소에 r1 레지스터의 값을 저장하는 연산이다.

d) What is the meaning of the instruction? : r1 레지스터의 값을 메모리의 (r3 + 0x000) 주소에 저장한다.

Address 017 | Instruction E59F9E98

a) Change to binary format: 1110 0101 1001 1111 1001 1110 1001 1000
b) Write assembly code: LDR r9, [pc, #+0xE98]
c) Describe why you wrote the assembly code like above:
    a. Figure A3-1 에 따르면, [25:27] 비트가 '010' 이므로 Load/store immediate offset instruction 이다.
    b. Cond field 의 값이 1110 (AL) 이므로 조건 없이 실행된다.
    c. Figure 4-14 에 따르면, L 비트가 '1' 이므로 Load instruction 이다.
    d. Load instuction 의 Syntax 는 LDR {<cond>} <Rd> <Addressing_mode> 이다.
    e. A 5.2.1 에 따르면, P == 1, W == 0 이므로 base register 의 값은 업데이트되지 않는다.
    f. Rn == 1111 (r15), Rd == 1001 (r9), offset_12 == 1110 1001 1000 (0xE98)이다.
    g. A5.2.2 에 따르면, U == 1 이므로 address 는 Rn 레지스터에 들어있는 주소값에 offset_12 를 더한 값이다.
    h. 즉, 메모리의 (r15 + 0xE98) 주소에 있는 값을 r9 레지스터에 저장하는 연산이다.
    i. 그런데 r15 레지스터는 pc 값을 저장하므로 pc + 0xE98 이다.
d) What is the meaning of the instruction? : 메모리의 (pc + 0xE98) 주소에 있는 값을 r9 레지스터에 저장한다.

Address 018 | Instruction E3A08000

a) Change to binary format: 1110 0011 1010 0000 1000 0000 0000 0000
b) Write assembly code: MOV r8, #0x00
c) Describe why you wrote the assembly code like above:
    a. Figure A3-1 에 따르면, [25:27]비트가 '001'이고, [23:24]비트가 '10'이 아니므로 Data processing instruction 이다. 그중에서도 opcode 가 '1101'이므로 MOV instruction 이다.
    b. MOV instruction 의 syntax 는 MOV {<cond>}{S} <Rd>, <shifter_operand> 이다.
    c. Cond field 의 값이 '1110'이므로 조건 없이 실행된다.
    d. A4.1.35 에 따르면, I == 1 이므로 Immediate value 를 사용한다.
    e. A4.1.35 에 따르면, S==0 이므로 MOV 명령어이고, CPSR 을 업데이트하지 않는다.

f. Rd == 1000 (r8)
g. A5.1.3 에 따르면, rotate_imm == 0000 이고 immed_8 은 0000 0000 (0x00) 이므로 <immediate> = 0x00, shifter_carry_out = C flag 이다.
h. 따라서 r8 레지스터에 상수 0x00 을 복사하는 연산이다.

d) What is the meaning of the instruction? : r8 레지스터에 상수 0x00 을 복사한다.

Address 019 | Instruction E5898000

a) Change to binary format: 1110 0101 1000 1001 1000 0000 0000 0000
b) Write assembly code: STR r8, [r9, #+0x000]
c) Describe why you wrote the assembly code like above:
   a. Figure A3-1 에 따르면, [25:27] 비트가 '010' 이므로 Load/store immediate offset instruction 이다.
   b. Cond field 의 값이 1110 (AL) 이므로 조건 없이 실행된다.
   c. Figure 4-14 에 따르면, L 비트가 '0' 이므로 Store instruction 이다.
   d. Store instuction 의 Syntax 는 STR {<cond>} <Rd> <Addressing_mode> 이다.
   e. A 5.2.1 에 따르면, P == 1, W == 0 이므로 base register 의 값은 업데이트되지 않는다.
   f. Rn == 1001 (r9), Rd == 1000 (r8), offset_12 == 0000 0000 0000 (0x000)이다.
   g. A5.2.2 에 따르면, U == 1 이므로 address 는 Rn 레지스터에 들어있는 주소값에 offset_12 를 더한 값이다.
   h. 즉, 메모리의 (r9 + 0x000) 번째 주소에 r8 레지스터의 값을 저장하는 연산이다.
d) What is the meaning of the instruction? : r8 레지스터의 값을 메모리의 (r9 + 0x000) 주소에 저장한다.

Address 01A | Instruction E5898004

a) Change to binary format: 1110 0101 1000 1001 1000 0000 0000 0100
b) Write assembly code: STR r8, [r9, #+0x004]
c) Describe why you wrote the assembly code like above:
   a. Figure A3-1 에 따르면, [25:27] 비트가 '010' 이므로 Load/store immediate offset instruction 이다.
   b. Cond field 의 값이 1110 (AL) 이므로 조건 없이 실행된다.
   c. Figure 4-14 에 따르면, L 비트가 '0' 이므로 Store instruction 이다.
   d. Store instuction 의 Syntax 는 STR {<cond>} <Rd> <Addressing_mode> 이다.

e. A 5.2.1 에 따르면, P == 1, W == 0 이므로 base register 의 값은 업데이트되지 않는다.
f. Rn == 1001 (r9), Rd == 1000 (r8), offset_12 == 0000 0000 0004 (0x004)이다.
g. A5.2.2 에 따르면, U == 1 이므로 address 는 Rn 레지스터에 들어있는 주소값에 offset_12 를 더한 값이다.
h. 즉, 메모리의 (r9 + 0x004) 번째 주소에 r8 레지스터의 값을 저장하는 연산이다.

d) What is the meaning of the instruction? : r8 레지스터의 값을 메모리의 (r9 + 0x004) 주소에 저장한다.

Address 01B | Instruction E5898008

a) Change to binary format: 1110 0101 1000 1001 1000 0000 0000 1000
b) Write assembly code: STR r8, [r9, #+0x008]
c) Describe why you wrote the assembly code like above:
a. Figure A3-1 에 따르면, [25:27] 비트가 '010' 이므로 Load/store immediate offset instruction 이다.
b. Cond field 의 값이 1110 (AL) 이므로 조건 없이 실행된다.
c. Figure 4-14 에 따르면, L 비트가 '0' 이므로 Store instruction 이다.
d. Store instuction 의 Syntax 는 STR {<cond>} <Rd> <Addressing_mode> 이다.
e. A 5.2.1 에 따르면, P == 1, W == 0 이므로 base register 의 값은 업데이트되지 않는다.
f. Rn == 1001 (r9), Rd == 1000 (r8), offset_12 == 0000 0000 0008 (0x008)이다.
g. A5.2.2 에 따르면, U == 1 이므로 address 는 Rn 레지스터에 들어있는 주소값에 offset_12 를 더한 값이다.
h. 즉, 메모리의 (r9 + 0x008) 번째 주소에 r8 레지스터의 값을 저장하는 연산이다.

d) What is the meaning of the instruction? : r8 레지스터의 값을 메모리의 (r9 + 0x008) 주소에 저장한다.

Address 01C | Instruction E589800C

a) Change to binary format: 1110 0101 1000 1001 1000 0000 0000 1100
b) Write assembly code: STR r8, [r9, #+0x00C]
c) Describe why you wrote the assembly code like above:
a. Figure A3-1 에 따르면, [25:27] 비트가 '010' 이므로 Load/store immediate offset instruction 이다.
b. Cond field 의 값이 1110 (AL) 이므로 조건 없이 실행된다.

c. Figure 4-14 에 따르면, L 비트가 '0' 이므로 Store instruction 이다.
d. Store instuction 의 Syntax 는 STR {<cond>} <Rd> <Addressing_mode> 이다.
e. A 5.2.1 에 따르면, P == 1, W == 0 이므로 base register 의 값은 업데이트되지 않는다.
f. Rn == 1001 (r9), Rd == 1000 (r8), offset_12 == 0000 0000 1100 (0x00C)이다.
g. A5.2.2 에 따르면, U == 1 이므로 address 는 Rn 레지스터에 들어있는 주소값에 offset_12 를 더한 값이다.
h. 즉, 메모리의 (r9 + 0x00C) 번째 주소에 r8 레지스터의 값을 저장하는 연산이다.

d) What is the meaning of the instruction? : r8 레지스터의 값을 메모리의 (r9 + 0x00C) 주소에 저장한다.

Address 01D | Instruction E5898010

a) Change to binary format: 1110 0101 1000 1001 1000 0000 0001 0000
b) Write assembly code: STR r8, [r9, #+0x010]
c) Describe why you wrote the assembly code like above:
   a. Figure A3-1 에 따르면, [25:27] 비트가 '010' 이므로 Load/store immediate offset instruction 이다.
   b. Cond field 의 값이 1110 (AL) 이므로 조건 없이 실행된다.
   c. Figure 4-14 에 따르면, L 비트가 '0' 이므로 Store instruction 이다.
   d. Store instuction 의 Syntax 는 STR {<cond>} <Rd> <Addressing_mode> 이다.
   e. A 5.2.1 에 따르면, P == 1, W == 0 이므로 base register 의 값은 업데이트되지 않는다.
   f. Rn == 1001 (r9), Rd == 1000 (r8), offset_12 == 0000 0001 0000 (0x010)이다.
   g. A5.2.2 에 따르면, U == 1 이므로 address 는 Rn 레지스터에 들어있는 주소값에 offset_12 를 더한 값이다.
   h. 즉, 메모리의 (r9 + 0x010) 번째 주소에 r8 레지스터의 값을 저장하는 연산이다.
d) What is the meaning of the instruction? : r8 레지스터의 값을 메모리의 (r9 + 0x010) 주소에 저장한다.

Address 01E | Instruction E5898014

a) Change to binary format: 1110 0101 1000 1001 1000 0000 0001 0100
b) Write assembly code: STR r8, [r9, #+0x014]
c) Describe why you wrote the assembly code like above:

a. Figure A3-1 에 따르면, [25:27] 비트가 '010' 이므로 Load/store immediate offset instruction 이다.
b. Cond field 의 값이 1110 (AL) 이므로 조건 없이 실행된다.
c. Figure 4-14 에 따르면, L 비트가 '0' 이므로 Store instruction 이다.
d. Store instuction 의 Syntax 는 STR {<cond>} <Rd> <Addressing_mode> 이다.
e. A 5.2.1 에 따르면, P == 1, W == 0 이므로 base register 의 값은 업데이트되지 않는다.
f. Rn == 1001 (r9), Rd == 1000 (r8), offset_12 == 0000 0001 0100 (0x014)이다.
g. A5.2.2 에 따르면, U == 1 이므로 address 는 Rn 레지스터에 들어있는 주소값에 offset_12 를 더한 값이다.
h. 즉, 메모리의 (r9 + 0x014) 번째 주소에 r8 레지스터의 값을 저장하는 연산이다.

d) What is the meaning of the instruction? : r8 레지스터의 값을 메모리의 (r9 + 0x014) 주소에 저장한다.

Address 01F | Instruction E5898018

a) Change to binary format: 1110 0101 1000 1001 1000 0000 0001 1000
b) Write assembly code: STR r8, [r9, #+0x018]
c) Describe why you wrote the assembly code like above:
   a. Figure A3-1 에 따르면, [25:27] 비트가 '010' 이므로 Load/store immediate offset instruction 이다.
   b. Cond field 의 값이 1110 (AL) 이므로 조건 없이 실행된다.
   c. Figure 4-14 에 따르면, L 비트가 '0' 이므로 Store instruction 이다.
   d. Store instuction 의 Syntax 는 STR {<cond>} <Rd> <Addressing_mode> 이다.
   e. A 5.2.1 에 따르면, P == 1, W == 0 이므로 base register 의 값은 업데이트되지 않는다.
   f. Rn == 1001 (r9), Rd == 1000 (r8), offset_12 == 0000 0001 1000 (0x018)이다.
   g. A5.2.2 에 따르면, U == 1 이므로 address 는 Rn 레지스터에 들어있는 주소값에 offset_12 를 더한 값이다.
   h. 즉, 메모리의 (r9 + 0x018) 번째 주소에 r8 레지스터의 값을 저장하는 연산이다.
d) What is the meaning of the instruction? : r8 레지스터의 값을 메모리의 (r9 + 0x018) 주소에 저장한다.

Address 020 | Instruction E59FDE78

a) Change to binary format: 1110 0101 1001 1111 1101 1110 0111 1000
b) Write assembly code: LDR r13, [pc, #+0xE78]
c) Describe why you wrote the assembly code like above:
    a. Figure A3-1 에 따르면, [25:27] 비트가 '010' 이므로 Load/store immediate offset instruction 이다.
    b. Cond field 의 값이 1110 (AL) 이므로 조건 없이 실행된다.
    c. Figure 4-14 에 따르면, L 비트가 '1' 이므로 Load instruction 이다.
    d. Load instuction 의 Syntax 는 LDR {<cond>} <Rd> <Addressing_mode> 이다.
    e. A 5.2.1 에 따르면, P == 1, W == 0 이므로 base register 의 값은 업데이트되지 않는다.
    f. Rn == 1111 (r15), Rd == 1101 (r13), offset_12 == 1110 0111 1000 (0xE78)이다.
    g. A5.2.2 에 따르면, U == 1 이므로 address 는 Rn 레지스터에 들어있는 주소값에 offset_12 를 더한 값이다.
    h. 즉, 메모리의 (r15 + 0xE78) 주소에 있는 값을 r13 레지스터에 저장하는 연산이다.
    i. 그런데 r15 레지스터는 pc 값을 저장하므로 pc + 0xE78 이다.
d) What is the meaning of the instruction? : 메모리의 (pc + 0xE78) 주소에 있는 값을 r13 레지스터에 저장한다.

Address 021 | Instruction E5931200

a) Change to binary format: 1110 0101 1001 0011 0001 0010 0000 0000
b) Write assembly code: LDR r1, [r3, #+0x200]
c) Describe why you wrote the assembly code like above:
    a. Figure A3-1 에 따르면, [25:27] 비트가 '010' 이므로 Load/store immediate offset instruction 이다.
    b. Cond field 의 값이 1110 (AL) 이므로 조건 없이 실행된다.
    c. Figure 4-14 에 따르면, L 비트가 '1' 이므로 Load instruction 이다.
    d. Load instuction 의 Syntax 는 LDR {<cond>} <Rd> <Addressing_mode> 이다.
    e. A 5.2.1 에 따르면, P == 1, W == 0 이므로 base register 의 값은 업데이트되지 않는다.
    f. Rn == 0011 (r3), Rd == 0001 (r1), offset_12 == 0010 0000 0000 (0x200)이다.
    g. A5.2.2 에 따르면, U == 1 이므로 address 는 Rn 레지스터에 들어있는 주소값에 offset_12 를 더한 값이다.
    h. 즉, 메모리의 (r3 + 0x200) 주소에 있는 값을 r1 레지스터에 저장하는 연산이다.

d) What is the meaning of the instruction? : 메모리의 (r3 + 0x200) 주소에 있는 값을 r1 레지스터에 저장한다.

Address 022 | Instruction E3510001

a) Change to binary format: 1110 0011 0101 0001 0000 0000 0000 0001
b) Write assembly code: CMP r1, #0x01
c) Describe why you wrote the assembly code like above:
    a. Figure A3-1 에 따르면, [25:27]비트가 '001'이므로 Data processing immediate instruction 이고, 그 중에서도 opcode 가 '1010'이므로 CMP instruction 이다.
    b. Cond field 의 값이 '1110' (AL)이므로 조건 없이 실행된다.
    c. CMP instruction 의 syntax 는 CMP{<cond>} <Rn>, <shifter_operand>이다.
    d. A4.1.15 에 따르면, Rn == 0001 (r1)
    e. A5.1.1 에 따르면, 25 번째 비트가 1 이므로 shifter_operand 는 immediate value 이다.
    f. rotate_imm == 0000, immed_8 == 0000 0001 (0x01)이므로 <immediate> = 0x01 이다.
    g. A4.1.15 에 따르면, Rn – shifter_operand == 0 이면 Z flag 를 1 로, 0 이 아니면 Z flag 를 0 으로 업데이트한다. 즉, r1 에 들어있는 값이 0x01 이면 Z 를 1 로, 아니면 0 으로 업데이트한다. 또한, Rn – shifter_operand 의 값이 음수라면 N flag 가 set 된다.
d) What is the meaning of the instruction? : r1 의 값이 1 이면 Z set, 아니면 Z clear

Address 023 | Instruction 0A000000

a) Change to binary format: 0000 1010 0000 0000 0000 0000 0000 0000
b) Write assembly code: BEQ #025
c) Describe why you wrote the assembly code like above:
    a. Figure A3-1 에 따르면, [25:27] 비트가 '101' 이므로 branch instruction 이다.
    b. Cond field 의 값이 0000 (EQ)이므로 Z 가 set 되어있는 상태일 때 실행된다.
    c. Branch instruction 의 syntax 는 B{L}{cond} <target address> 이다.
    d. A4.1.5 에 따르면, L 비트가 0 이므로 BEQ instruction 이다.
    e. A4.1.5 에 따르면, <target address> 는 다음과 같이 계산된다.
        i. 24-bit immediate 0000 0000 0000 0000 0000 0000 을 30-bit 로 sign-extending 한 결과 00 0000 0000 0000 0000 0000 0000 을 얻는다.
        ii. 얻은 결과를 2-bit shift left 하면
        0000 0000 0000 0000 0000 0000 0000 0000 = 0 (base 10)

iii. PC 값에 이 값을 더해 target address 를 얻는다.
{(35*4 + 8) + (0)} / 4 = 37 (base 10) = 0x025 (base 16)
iv. 따라서 assembly code 는 BEQ #025 이다.

d) What is the meaning of the instruction? : Z flag 가 set 되어 있다면 address 025 로 branch 한다.

Address 024 | Instruction EAFFFFFB

a) Change to binary format: 1110 1010 1111 1111 1111 1111 1111 1011
b) Write assembly code: B #021
c) Describe why you wrote the assembly code like above:
   a. Figure A3-1 에 따르면, [25:27] 비트가 '101' 이므로 branch instruction 이다.
   b. Cond field 의 값이 1110 (AL) 이므로 조건 없이 실행된다.
   c. Branch instruction 의 syntax 는 B{L}{cond} <target address> 이다.
   d. A4.1.5 에 따르면, L 비트가 0 이므로 B instruction 이다.
   e. A4.1.5 에 따르면, <target address> 는 다음과 같이 계산된다.
      i. 24-bit immediate 1111 1111 1111 1111 1111 1011 을 30-bit 로 sign-extending 한 결과 11 1111 1111 1111 1111 1111 1111 1011 을 얻는다.
      ii. 얻은 결과를 2-bit shift left 하면
      1111 1111 1111 1111 1111 1111 1110 1100 = -20 (base 10)
      iii. PC 값에 이 값을 더해 target address 를 얻는다.
      {(36*4 + 8) + (-20)} / 4 = 33 (base 10) = 0x021 (base 16)
      iv. 따라서 assembly code 는 B #021 이다.
d) What is the meaning of the instruction? : address 021 로 branch 한다.

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

1. 000: B #008 에 의해 address 008 로 branch
2. 008: LDR r2, [pc, #0xEC8]에서 pc 값은 8*4+8 = 40 (base 10) = 0x028 (base 16) 이므로 0x028 에 0xEC8 을 더한 0xEF0 에서 워드를 읽어 r2 에 저장. 이때, 메모리의 0xEF0 주소에는 0x100 이란 값이 있었다고 임의로 가정. 즉, r2 = 0x100
3. 009: MOV r0, #0x40 에 의해 r0 레지스터에 상수 0x40 저장. 즉, r0 = 0x40
4. 00A ~ 00F: STR 명령어에 의해 메모리의 0x110 ~ 0x124 주소에 0x40 저장(4 바이트 단위). 즉, Mem[0x110] = Mem[0x114] = … = Mem[0x124] = 0x40
5. 010: MOV r0, #0x3F 에 의해 r0 레지스터의 값 0x3F 로 변경

6. 011: STR r0, [r2, #0x028] -> Mem[0x128] = 0x3F
7. 012: MOV r0, #0x08 에 의해 r0 레지스터의 값 0x08 로 변경
8. 013: STR r0, [r2, #0x02C] -> Mem[0x12C] = 0x08
9. 014: LDR r3, [pc, #0xE9C]에서 pc 값은 20*4+8 = 0x058. 이 값에 0xE9C 를 더한 0xEF4 주소에 있는 값을 읽어 r3 에 저장. 이때, Mem[0xEF4] = 0x500 이었다고 임의로 가정. 즉, r3 = 0x500
10. 015: LDR r1, [pc, #0xE9C]위와 마찬가지로, 메모리의 0xEF8 주소에 있는 값을 읽어 r1 에 저장. 이때, Mem[0xEF8] = 0x600 이었다고 임의로 가정. 즉, r1 = 0x600
11. 016: STR r1, [r3, #0x000]에서 메모리의 r3 주소에 r1 저장. 이때 r3 = 0x500 이고, r1 = 0x600 이었으므로 Mem[0x500] = 0x600
12. 017: LDR r9, [pc, #0xE98]에서 pc 값은 23*4 + 8 = 0x064. 이 값에 0xE98 을 더한 0xEFC 주소에 있는 값을 읽어 r9 에 저장. 이때, Mem[0xEFC] = 0x800 이었다고 임의로 가정. 즉, r9 = 0x800
13. 018: MOV r8, #0x00 -> r8 = 0x00
14. 019 ~ 01F: STR 명령어에 의해 메모리의 0x800 ~ 0x818 주소에 r8 의 값 저장. 이때 r8 에 들어있던 값은 0x00 이라고 임의로 가정. 즉, Mem[0x800] = Mem[0x804] = … = Mem[0x818] = 0x00
15. 020: LDR r13, [pc, #0xE78]에서 pc 값은 32*4 + 8 = 0x088. 이 값에 0xE78 을 더한 0xF00 주소에 있는 값을 읽어 r13 에 저장. 이때, Mem[0xF00] = 0xA00 이었다고 임의로 가정. 즉, r13 = 0xA00
16. 021: LDR r1, [r3, #0x200]에서 r3=0x500 이었으므로 메모리의 0x700 주소에서 값을 읽어 r1 레지스터에 저장. 즉, r1 = Mem[0x700]
17. 022: CMP r1, #0x01 에서 r1 의 값이 0x01 이라면 Z set. 아닐 경우 set 하지 않음.
18. 023: BEQ #025
    A. 만약 위에서 r1 의 값이 0x01 이었다면 Z set 이므로 Address 025 로 이동해 프로그램 종료.
    B. r1 의 값이 0x01 이 아니었다면 Address 024 로 이동.
        i. 024: B #021 에서 Address 021 로 다시 이동하지만, r1 의 값은 0x01 이 아니었으므로 021 ~ 024 에서 무한루프에 빠지게 됨.

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

014: LDR r3, [pc, #0xE9C]에서 r3 = Mem[0xEF4]이고,

021: LDR r1, [r3, #0x200]에서 r1 = Mem[r3 + 0x200]이므로 r1 = Mem[Mem[0xEF4] +0x200]

그런데 022~023 에서 만약 r1 의 값이 1 이 아니라면 021~024 에서 무한루프에 빠지게 되므로 Mem[Mem[0xEF4] + 0x200]에 들어있던 값에 따라 execution 이 끝날지 아닐지 결정된다.

1) Mem[Mem[0xEF4] + 0x200]의 값이 1 인 경우 => 023 에서 025 로 branch 하며 종료.
2) Mem[Mem[0xEF4] + 0x200]의 값이 1 이 아닌 경우 => 021~024 에서 무한루프.