**Computer Architecture Assignment #1**

Your Student Number: WRITE YOUR STUDENT NUMBER

Name: WRITE YOUR NAME

<<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)**

1. Change to binary format: 1110 1010 0000 0000 0000 0000 0000 0110
2. Write assembly code: B #8;
3. Describe why you wrote the assembly code like above:
   1. 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.
   2. 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.
   3. 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.
   4. Operation – Target Address: According to page A4-10 in ARM manual, the target address is calculated like below.
      1. First, the result of sign-extending the 24-bit signed immediate to 30 bits is 00 0000 0000 0000 0000 0000 0000 0000 0000 0110. (Because the signed immediate is 0000 0000 0000 0000 0000 0110 here.)
      2. Then, get 0000 0000 0000 0000 0000 0000 0001 1000 by shifting the result left two bits.
      3. 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.
      4. 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>’.
4. What is the meaning of the instruction? : The instruction means ‘branch to address 8’.

Address 001 | Instruction EAFFFFFE

1. Change to binary format: 1110 1010 111111111111111111111110
2. Write assembly code: B #-2
3. Describe why you wrote the assembly code like above:
   1. Condition : 1110 (always)
   2. Operator: Branch
   3. L: 0
   4. Offset : -2
4. What is the meaning of the instruction? :
   1. 계산
      1. Add # = 1
      2. PC = add#\*4 + 8 = 12
      3. New PC = PC + offset\*4 = 4
      4. New Addr # = new PC/4 = 1
   2. 작동
      1. # 1 로 branch
      2. 무한 loop

Address 002 | Instruction EA0000A7

1. Change to binary format: 1110 1010 000000000000000010100111
2. Write assembly code: B #167
3. Describe why you wrote the assembly code like above:
   1. Condition : 1110 (always)
   2. Operator: Branch
   3. L: 0
   4. Offset : A7 = 167
4. What is the meaning of the instruction? :
   1. 계산
      1. Add # = 2
      2. PC = add#\*4 + 8 = 16
      3. New PC = PC + offset\*4 = 684
      4. New Addr # = new PC/4 = 171
   2. 작동
      1. # 171 로 branch

Address 003~005 | Instruction EAFFFFFE

1. Change to binary format: 1110 1010 111111111111111111111110
2. Write assembly code: B#-2
3. Describe why you wrote the assembly code like above:
   1. 계산
      1. Add # = self
      2. PC = add#\*4 + 8 = self\*4 + 8
      3. New PC = PC + offset\*4 = (self\*4 + 8) -8 = self\*4
      4. New Addr # = new PC/4 = self
4. What is the meaning of the instruction? :
   1. # self 로 branch

Address 006 | Instruction EA0000A4

1. Change to binary format: 1110 1010 000000000000000010100100
2. Write assembly code: B #164
3. Describe why you wrote the assembly code like above:
   1. Condition : 1110 (always)
   2. Operator: Branch
   3. L: 0
   4. Offset : A7 = 164
4. What is the meaning of the instruction? :
   1. 계산
      1. Add # = 6
      2. PC = add#\*4 + 8 = 32
      3. New PC = PC + offset\*4 = 688
      4. New Addr # = new PC/4 = 172
   2. 작동
      1. # 172 로 branch

Address 007 | Instruction EAFFFFFE

1. Change to binary format: 1110 1010 111111111111111111111110
2. Write assembly code: B #-2
3. Describe why you wrote the assembly code like above:
   1. Condition : 1110 (always)
   2. Operator: Branch
   3. L: 0
   4. Offset : -2
4. What is the meaning of the instruction? :
   1. 계산
      1. Add # = 7
      2. PC = add#\*4 + 8 = 36
      3. New PC = PC + offset\*4 = 28
      4. New Addr # = new PC/4 = 7
   2. 작동
      1. # 7 로 branch
      2. 무한 loop

Address 008 | Instruction E59F2EC8

1. Change to binary format: 1110 0101 1001 1111 0010 111011001000
2. Write assembly code: LDR $2, [$15, #0xEC8]
3. Describe why you wrote the assembly code like above:
   1. Condition : 1110 (always)
   2. Operator: single data transfer
   3. IPUBWL: 011001
   4. Rn: 15, Rd: 2
   5. Offset : #0xEC8
4. What is the meaning of the instruction? :
   1. 계산
      1. Address = Reg[15] + #0xEC8
      2. Reg[2] = mem[Address]
   2. 작동
      1. $15 + #0xEC8 주소의 메모리에서 값을 가져와 $2에 저장

Address 009 | Instruction E3A00040

1. Change to binary format: 1110 0011 1010 0000 0000 000001000000
2. Write assembly code: MOV $0 #0x40
3. Describe why you wrote the assembly code like above:
   1. Condition : 1110 (always)
   2. Operator: data processing
   3. I: 1, Opcode: 1101, S:0
   4. Rn: 0000, Rd: 0000
   5. Operand: #0x40
4. What is the meaning of the instruction? :
   1. 계산
      1. Reg[0] = #0x40
   2. 작동
      1. $0에 #0x40저장

Address 00A | Instruction E5820010

1. Change to binary format: 1110 0101 1000 0010 0000 0000 0001 0000
2. Write assembly code: STR $0, [$2, #0x10]
3. Describe why you wrote the assembly code like above:
   1. Condition : 1110 (always)
   2. Operator: single data transfer
   3. IPUBWL: 011000
   4. Rn: 2, Rd: 0
   5. Offset : #0x10
4. What is the meaning of the instruction? :
   1. 계산
      1. Address = Reg[2] + #0x10
      2. mem[Address] = Reg[0]
   2. 작동
      1. $2 + #0x10의 메모리에 $0 에 있는 값을 저장
5. :

Address 00B | Instruction E5820014

1. Change to binary format: 1110 0101 1000 0010 0000 0000 0001 0100
2. Write assembly code: STR $0, [$2, #0x14]
3. Describe why you wrote the assembly code like above:
   1. Condition : 1110 (always)
   2. Operator: single data transfer
   3. IPUBWL: 011000
   4. Rn: 2, Rd: 0
   5. Offset : #0x14
4. What is the meaning of the instruction? :
   1. 계산
      1. Address = Reg[2] + #0x14
      2. mem[Address] = Reg[0]
   2. 작동
      1. $2 + #0x14의 메모리에 $0 에 있는 값을 저장

Address 00C | Instruction E5820018

1. Change to binary format: 1110 0101 1000 0010 0000 0000 0001 1000
2. Write assembly code: STR $0, [$2, #0x18]
3. Describe why you wrote the assembly code like above:
   1. Condition : 1110 (always)
   2. Operator: single data transfer
   3. IPUBWL: 011000
   4. Rn: 2, Rd: 0
   5. Offset : #0x18
4. What is the meaning of the instruction? :
   1. 계산
      1. Address = Reg[2] + #0x18
      2. mem[Address] = Reg[0]
   2. 작동
      1. $2 + #0x18의 메모리에 $0 에 있는 값을 저장

Address 00D | Instruction E582001C

1. Change to binary format: 1110 0101 1000 0010 0000 0000 0001 1100
2. Write assembly code: STR $0, [$2, #0x1C]
3. Describe why you wrote the assembly code like above:
   1. Condition : 1110 (always)
   2. Operator: single data transfer
   3. IPUBWL: 011000
   4. Rn: 2, Rd: 0
   5. Offset : #0x1C
4. What is the meaning of the instruction? :
   1. 계산
      1. Address = Reg[2] + #0x1C
      2. mem[Address] = Reg[0]
   2. 작동
      1. $2 + #0x1C 의 메모리에 $0 에 있는 값을 저장

Address 00E | Instruction E5820020

1. Change to binary format: 1110 0101 1000 0010 0000 0000 0010 0000
2. Write assembly code: STR $0, [$2, #0x20]
3. Describe why you wrote the assembly code like above:
   1. Condition : 1110 (always)
   2. Operator: single data transfer
   3. IPUBWL: 011000
   4. Rn: 2, Rd: 0
   5. Offset : #0x20
4. What is the meaning of the instruction? :
   1. 계산
      1. Address = Reg[2] + #0x20
      2. mem[Address] = Reg[0]
   2. 작동
      1. $2 + #0x20 의 메모리에 $0 에 있는 값을 저장

Address 00F | Instruction E5820024

1. Change to binary format: 1110 0101 1000 0010 0000 0000 0010 0100
2. Write assembly code: STR $0, [$2, #0x24]
3. Describe why you wrote the assembly code like above:
   1. Condition : 1110 (always)
   2. Operator: single data transfer
   3. IPUBWL: 011000
   4. Rn: 2, Rd: 0
   5. Offset : #0x24
4. What is the meaning of the instruction? :
   1. 계산
      1. Address = Reg[2] + #0x24
      2. mem[Address] = Reg[0]
   2. 작동
      1. $2 + #0x24 의 메모리에 $0 에 있는 값을 저장

Address 010 | Instruction E3A0003F

1. Change to binary format: 1110 0011 1010 0000 0000 0000 0011 1111
2. Write assembly code: MOV $0 #0x3F
3. Describe why you wrote the assembly code like above:
   1. Condition : 1110 (always)
   2. Operator: data processing
   3. I: 1, Opcode: 1101, S:0
   4. Rn: 0000, Rd: 0000
   5. Operand: #0x3F
4. What is the meaning of the instruction? :
   1. 계산
      1. Reg[0] = #0x3F
   2. 작동
      1. $0에 #0x3F 저장

Address 011 | Instruction E5820028

1. Change to binary format: 1110 0101 1000 0010 0000 0000 0010 1000
2. Write assembly code: STR $0, [$2, #0x28]
3. Describe why you wrote the assembly code like above:
   1. Condition : 1110 (always)
   2. Operator: single data transfer
   3. IPUBWL: 011000
   4. Rn: 2, Rd: 0
   5. Offset : #0x28
4. What is the meaning of the instruction? :
   1. 계산
      1. Address = Reg[2] + #0x28
      2. mem[Address] = Reg[0]
   2. 작동
      1. $2 + #0x28 의 메모리에 $0 에 있는 값을 저장
5. :

Address 012 | Instruction E3A00008

1. Change to binary format: 1110 0011 1010 0000 0000 0000 0000 1000
2. Write assembly code: MOV $0 #0x08
3. Describe why you wrote the assembly code like above:
   1. Condition : 1110 (always)
   2. Operator: data processing
   3. I: 1, Opcode: 1101, S:0
   4. Rn: 0000, Rd: 0000
   5. Operand: #0x08
4. What is the meaning of the instruction? :
   1. 계산
      1. Reg[0] = #0x08
   2. 작동
      1. $0에 #0x08 저장

Address 013 | Instruction E582002C

1. Change to binary format: 1110 0101 1000 0010 0000 0000 0010 1100
2. Write assembly code: STR $0, [$2, #0x2C]
3. Describe why you wrote the assembly code like above:
   1. Condition : 1110 (always)
   2. Operator: single data transfer
   3. IPUBWL: 011000
   4. Rn: 2, Rd: 0
   5. Offset : #0x2C
4. What is the meaning of the instruction? :
   1. 계산
      1. Address = Reg[2] + #0x2C
      2. mem[Address] = Reg[0]
   2. 작동
      1. $2 + #0x2C 의 메모리에 $0 에 있는 값을 저장

Address 014 | Instruction E59F3E9C

1. Change to binary format: 1110 0101 1001 1111 0011 1110 1001 1100
2. Write assembly code: LDR $3, [$15, #0xE9C]
3. Describe why you wrote the assembly code like above:
   1. Condition : 1110 (always)
   2. Operator: single data transfer
   3. IPUBWL: 011001
   4. Rn: 15, Rd: 3
   5. Offset : #0xE9C
4. What is the meaning of the instruction? :
   1. 계산
      1. Address = Reg[15] + #0xE9C
      2. Reg[3] = mem[Address]
   2. 작동
      1. $15 + #0xE9C 주소의 메모리에서 값을 가져와 $3에 저장

Address 015 | Instruction E59F1E9C

1. Change to binary format: 1110 0101 1001 1111 0001 1110 1001 1100
2. Write assembly code: LDR $1, [$15, #0xE9C]
3. Describe why you wrote the assembly code like above:
   1. Condition : 1110 (always)
   2. Operator: single data transfer
   3. IPUBWL: 011001
   4. Rn: 15, Rd: 1
   5. Offset : #0xE9C
4. What is the meaning of the instruction? :
   1. 계산
      1. Address = Reg[15] + #0xE9C
      2. Reg[1] = mem[Address]
   2. 작동
      1. $15 + #0xE9C 주소의 메모리에서 값을 가져와 $1에 저장

Address 016 | Instruction E5831000

1. Change to binary format: 1110 0101 1000 0011 0001 0000 0000 0000
2. Write assembly code: STR $1, [$3, #0x000]
3. Describe why you wrote the assembly code like above:
   1. Condition : 1110 (always)
   2. Operator: single data transfer
   3. IPUBWL: 011000
   4. Rn: 3, Rd: 1
   5. Offset : #0x2C
4. What is the meaning of the instruction? :
   1. 계산
      1. Address = Reg[3] + #0x000
      2. mem[Address] = Reg[1]
   2. 작동
      1. $3 + #0x000 의 메모리에 $1 에 있는 값을 저장

Address 017 | Instruction E59F9E98

1. Change to binary format: 1110 0101 1001 1111 1001 1110 1001 1000
2. Write assembly code: LDR $9, [$15, #0xE98]
3. Describe why you wrote the assembly code like above:
   1. Condition : 1110 (always)
   2. Operator: single data transfer
   3. IPUBWL: 011001
   4. Rn: 15, Rd: 9
   5. Offset : #0xE98
4. What is the meaning of the instruction? :
   1. 계산
      1. Address = Reg[15] + #0xE98
      2. Reg[9] = mem[Address]
   2. 작동
      1. $15 + #0xE98 주소의 메모리에서 값을 가져와 $9에 저장

Address 018 | Instruction E3A08000

1. Change to binary format: 1110 0011 1010 0000 1000 0000 0000 0000
2. Write assembly code: MOV $8 #0x000
3. Describe why you wrote the assembly code like above:
   1. Condition : 1110 (always)
   2. Operator: data processing
   3. I: 1, Opcode: 1101, S:0
   4. Rn: 0, Rd: 8
   5. Operand: #0x000
4. What is the meaning of the instruction? :
   1. 계산
      1. Reg[8] = #0x000
   2. 작동
      1. $8에 #0x000저장

Address 019 | Instruction E5898000

1. Change to binary format: 1110 0101 1000 1001 1000 0000 0000 0000
2. Write assembly code: STR $8, [$9, #0x000]
3. Describe why you wrote the assembly code like above:
   1. Condition : 1110 (always)
   2. Operator: single data transfer
   3. IPUBWL: 011000
   4. Rn: 9, Rd: 8
   5. Offset : #0x000
4. What is the meaning of the instruction? :
   1. 계산
      1. Address = Reg[9] + #0x000
      2. mem[Address] = Reg[8]
   2. 작동
      1. $9 + #0x000 의 메모리에 $8 에 있는 값을 저장

Address 01A | Instruction E5898004

1. Change to binary format: 1110 0101 1000 1001 1000 0000 0000 0100
2. Write assembly code: STR $8, [$9, #0x004]
3. Describe why you wrote the assembly code like above:
   1. Condition : 1110 (always)
   2. Operator: single data transfer
   3. IPUBWL: 011000
   4. Rn: 9, Rd: 8
   5. Offset : #0x004
4. What is the meaning of the instruction? :
   1. 계산
      1. Address = Reg[9] + #0x004
      2. mem[Address] = Reg[8]
   2. 작동
      1. $9 + #0x004 의 메모리에 $8 에 있는 값을 저장

Address 01B | Instruction E5898008

1. Change to binary format: 1110 0101 1000 1001 1000 0000 0000 1000
2. Write assembly code: STR $8, [$9, #0x008]
3. Describe why you wrote the assembly code like above:
   1. Condition : 1110 (always)
   2. Operator: single data transfer
   3. IPUBWL: 011000
   4. Rn: 9, Rd: 8
   5. Offset : #0x008
4. What is the meaning of the instruction? :
   1. 계산
      1. Address = Reg[9] + #0x008
      2. mem[Address] = Reg[8]
   2. 작동
      1. $9 + #0x008 의 메모리에 $8 에 있는 값을 저장

Address 01C | Instruction E589800C

1. Change to binary format: 1110 0101 1000 1001 1000 0000 0000 1100
2. Write assembly code: STR $8, [$9, #0x00C]
3. Describe why you wrote the assembly code like above:
   1. Condition : 1110 (always)
   2. Operator: single data transfer
   3. IPUBWL: 011000
   4. Rn: 9, Rd: 8
   5. Offset : #0x00C
4. What is the meaning of the instruction? :
   1. 계산
      1. Address = Reg[9] + #0x00C
      2. mem[Address] = Reg[8]
   2. 작동
      1. $9 + #0x00C 의 메모리에 $8 에 있는 값을 저장

Address 01D | Instruction E5898010

1. Change to binary format: 1110 0101 1000 1001 1000 0000 0001 0000
2. Write assembly code: STR $8, [$9, #0x010]
3. Describe why you wrote the assembly code like above:
   1. Condition : 1110 (always)
   2. Operator: single data transfer
   3. IPUBWL: 011000
   4. Rn: 9, Rd: 8
   5. Offset : #0x010
4. What is the meaning of the instruction? :
   1. 계산
      1. Address = Reg[9] + #0x010
      2. mem[Address] = Reg[8]
   2. 작동
      1. $9 + #0x010 의 메모리에 $8 에 있는 값을 저장

Address 01E | Instruction E5898014

1. Change to binary format: 1110 0101 1000 1001 1000 0000 0001 0100
2. Write assembly code: STR $8, [$9, #0x014]
3. Describe why you wrote the assembly code like above:
   1. Condition : 1110 (always)
   2. Operator: single data transfer
   3. IPUBWL: 011000
   4. Rn: 9, Rd: 8
   5. Offset : #0x014
4. What is the meaning of the instruction? :
   1. 계산
      1. Address = Reg[9] + #0x014
      2. mem[Address] = Reg[8]
   2. 작동
      1. $9 + #0x014 의 메모리에 $8 에 있는 값을 저장

Address 01F | Instruction E5898018

1. Change to binary format: 1110 0101 1000 1001 1000 0000 0001 1000
2. Write assembly code: STR $8, [$9, #0x018]
3. Describe why you wrote the assembly code like above:
   1. Condition : 1110 (always)
   2. Operator: single data transfer
   3. IPUBWL: 011000
   4. Rn: 9, Rd: 8
   5. Offset : #0x018
4. What is the meaning of the instruction? :
   1. 계산
      1. Address = Reg[9] + #0x018
      2. mem[Address] = Reg[8]
   2. 작동
      1. $9 + #0x018 의 메모리에 $8 에 있는 값을 저장

Address 020 | Instruction E59FDE78

1. Change to binary format: 1110 0101 1001 1111 1101 1110 0111 1000
2. Write assembly code: LDR $13, [$15, #0xE78]
3. Describe why you wrote the assembly code like above:
   1. Condition : 1110 (always)
   2. Operator: single data transfer
   3. IPUBWL: 011001
   4. Rn: 15, Rd: 13
   5. Offset : #0xE78
4. What is the meaning of the instruction? :
   1. 계산
      1. Address = Reg[15] + #0xE78
      2. Reg[13] = mem[Address]
   2. 작동
      1. $15 + #0xE78 주소의 메모리에서 값을 가져와 $13 에 저장

Address 021 | Instruction E5931200

1. Change to binary format: 1110 0101 1001 0011 0001 0010 0000 0000
2. Write assembly code: LDR $1, [$3, #0x200]
3. Describe why you wrote the assembly code like above:
   1. Condition : 1110 (always)
   2. Operator: single data transfer
   3. IPUBWL: 011001
   4. Rn: 3, Rd: 1
   5. Offset : #0x200
4. What is the meaning of the instruction? :
   1. 계산
      1. Address = Reg[3] + #0x200
      2. Reg[1] = mem[Address]
   2. 작동
      1. $3 + #0x200 주소의 메모리에서 값을 가져와 $1에 저장

Address 022 | Instruction E3510001

1. Change to binary format: 1110 0011 0101 0001 0000 0000 0000 0001
2. Write assembly code: CMP $1 #0x01
3. Describe why you wrote the assembly code like above:
   1. Condition : 1110 (always)
   2. Operator: data processing
   3. I: 1, Opcode: 1010, S:1
   4. Rn: 1, Rd: 0
   5. Operand: #0x01
4. What is the meaning of the instruction? :
   1. 계산
      1. If Reg[1] == #0x01
         1. Z = 1
      2. Else
         1. Z = 0
         2. if Reg[1] < #0x01
            1. N = 1
         3. Else
            1. N = 0
   2. 작동
      1. $1 에 있는 값이 #0x01 과 같으면 Z flag를 1로 설정
      2. 다른경우 Z flag 를 0으로 설정
      3. 그리고Reg[1] 이 더 작으면 N = 1 로 설정
      4. 아닌경우 N = 0 으로 설정

Address 023 | Instruction 0A000000

1. Change to binary format: 0000 1010 0000 0000 0000 0000 0000 0000
2. Write assembly code: BEQ #0
3. Describe why you wrote the assembly code like above:
   1. Condition : 0000 (EQ)
   2. Operator: Branch
   3. L: 0
   4. Offset : 0x000
4. What is the meaning of the instruction? :
   1. 계산
      1. Add # = 23
      2. PC = add#\*4 + 8 = 100
      3. New PC = PC + offset\*4 = 100
      4. New Addr # = new PC/4 = 25
   2. 작동
      1. Z 가 1일 때 #25 로 branch
      2. Z 가 0 일 때 #24 로 branch

Address 024 | Instruction EAFFFFFB

1. Change to binary format: 1110 1010 1111 1111 1111 1111 1111 1011
2. Write assembly code: B #-5
3. Describe why you wrote the assembly code like above:
   1. Condition : 1110 (always)
   2. Operator: Branch
   3. L: 0
   4. Offset : -5
4. What is the meaning of the instruction? :
   1. 계산
      1. Add # = 24
      2. PC = add#\*4 + 8 = 104
      3. New PC = PC + offset\*4 = 84
      4. New Addr # = new PC/4 = 21
   2. 작동
      1. # 21 로 branch

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

* #000: #08로 이동
* #008: $2 ← mem[$15, #0xEC8]
* #009: $0 ← #0x40
* #00A~00F: $0 ← mem[$2,#0x010+i\*4]
* #010: $0 ← #0x3F
* #011: mem[$2, #0x28] ← $0
* #012: $0 ← #0x08
* #013: mem[$2, #0x2C] ← $0
* #014: $3 ← mem[$15, #0xE9C]
* #015: $1 ← mem[$15, #0xE9C]
* #016: mem[$3, #0x000] ← $1
* #017: $9 ← mem[$15, #0xE98]
* #018: $8 ← #0x000
* #019~01F: mem[$9, #0x000+i\*4] ← $8
* #020: $13 ← mem[$15, #0xE78]
* #021: $1 ← mem[$3, #0x200]
* #022: if $1 == #0x01 : Z=1 else Z=0
* #023: if Z == 1
  + #025:

Else

* + #024: #021로 이동

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

#022 의 $1 == #0x01 이 만족한다면 #025로 탈출할 것이고,

그렇지 않은경우 #021→ #022 → false → #023 → false → #024 → #021 의 loop 에 갇힌다.