Computer Architecture Preliminary Term Project

2017320108 컴퓨터 학과 고재영

000 : EA000006

1. 1110 1010 0000 0000 0000 0000 0000 0110
2. B #6
3. move to PC + 8 + 6x4 address
4. 따라서 PC + 4x8 만큼 가므로 008 주소에 있는 instruction 실행

008 : E59F2EC8

1. 1110 0101 1001 1111 0010 1110 1100 1000
2. LDR $2, [$15, #0xEC8]
3. I bit(25th bit) = 0 이므로 # 즉 immediate value 상수 값 의미
4. 레지스터 15 의 값과 immediate value #0xEC8 을 더해 (ALU op 인 add op 을 통해) 레지스터 2 에 메모리 주소값을 저장하는 로드 작업.
5. Branch 작업 없으므로 PC+4 즉 008 에서 009 주소에 있는 instruction 으로 넘어간다.

009 : E3A00040;

1. 1110 0011 1010 0000 0000 0000 0100 0000
2. MOV $0, #0x040
3. I bit = 1 이기 때문에 이는 immediate value 를 의미
4. 레지스터 0 에 상수 값 #0x040 을 저장한다.

00A : E5820010;

1. 1110 0101 1000 0010 0000 0000 0001 0000
2. STR $0, [$2, #0x010]
3. I bit = 0 이기 때문에 immediate value 를 의미
4. 레지스터 0 에 든 데이터를 레지스터 2 에 든 값과 상수값 #0x010 을 더한 메모리 주소에 저장.

00B : E5820014;

1. 1110 0101 1000 0010 0000 0000 0001 0100
2. STR $0, [$2, #0x014]
3. I bit = 0 이기 때문에 immediate value 를 의미
4. 레지스터 0 에 든 데이터를 레지스터 2 에 든 값과 상수값 #0x014 을 더한 메모리 주소에 저장.

00C : E5820018;

1. 1110 0101 1000 0010 0000 0000 0001 1000
2. STR $0, [$2, #0x018]

3. I bit = 0 이기 때문에 immediate value 를 의미
4. 레지스터 0 에 든 데이터를 레지스터 2 에 든 값과 상수값 #0x018 을 더한 메모리 주소에 저장.

00D : E582001C;

1. 1110 0101 1000 0010 0000 0000 0001 1100
2. STR $0, [$2, #0x01C]
3. I bit = 0 이기 때문에 immediate value 를 의미
4. 레지스터 0 에 든 데이터를 레지스터 2 에 든 값과 상수값 #0x01C 을 더한 메모리 주소에 저장.

00E : E5820020;

1. 1110 0101 1000 0010 0000 0000 0010 0000
2. STR $0, [$2, #0x020]
3. I bit = 0 이기 때문에 immediate value 를 의미
4. 레지스터 0 에 든 데이터를 레지스터 2 에 든 값과 상수값 #0x020 을 더한 메모리 주소에 저장.

00F : E5820024;

1. 1110 0101 1000 0010 0000 0000 0010 0100
2. STR $0, [$2, #0x024]
3. I bit = 0 이기 때문에 immediate value 를 의미
4. 레지스터 0 에 든 데이터를 레지스터 2 에 든 값과 상수값 #0x024 을 더한 메모리 주소에 저장.

010 : E3A0003F;

1. 1110 0011 1010 0000 0000 0000 0011 1111
2. MOV $0, #0x03F
3. I bit = 1 이기 때문에 이는 immediate value 를 의미
4. 레지스터 0 에 상수 값 #0x03F 을 저장한다.

011 : E5820028;

1. 1110 0101 1000 0010 0000 0000 0010 1000
2. STR $0, [$2, #0x028]
3. I bit = 0 이기 때문에 immediate value 를 의미
4. 레지스터 0 에 든 데이터를 레지스터 2 에 든 값과 상수값 #0x028 을 더한 메모리 주소에 저장.

012 : E3A00008;

1. 1110 0011 1010 0000 0000 0000 0000 1000
2. MOV $0, #0x008
3. I bit = 1 이기 때문에 이는 immediate value 를 의미
4. 레지스터 0 에 상수 값 #0x008 을 저장한다.

013 : E582002C;

1. 1110 0101 1000 0010 0000 0000 0010 1100
2. STR $0, [$2, #0x02C]
3. I bit = 0 이기 때문에 immediate value 를 의미
4. 레지스터 0 에 든 데이터를 레지스터 2 에 든 값과 상수값 #0x02C 을 더한 메모리 주소에 저장.

014 : E59F3E9C;

1. 1110 0101 1001 1111 0011 1110 1001 1100
2. LDR $3, [$15, #0xE9C]
3. I bit = 0 이므로 immediate value 상수 값 의미
4. 레지스터 15 의 값과 immediate value #0x E9C 을 더해 (ALU op 인 add op 을 통해) 레지스터 3 에 메모리 주소값을 저장하는 로드 작업.

015 : E59F1E9C;

1. 1110 0101 1001 1111 0001 1110 1001 1100
2. LDR $1, [$15, #0xE9C]
3. I bit = 0 이므로 # 즉 immediate value 상수 값 의미
4. 레지스터 15 의 값과 immediate value #0x E9C 을 더해 (ALU op 인 add op 을 통해) 레지스터 1 에 메모리 주소값을 저장하는 로드 작업.

016 : E5831000;

1. 1110 0101 1000 0011 0001 0000 0000 0000
2. STR $1, [$3, #0x000]
3. I bit = 0 이기 때문에 immediate value 를 의미
4. 레지스터 1 에 든 데이터를 레지스터 3 에 든 값과 상수값 #0x000 을 더한 메모리 주소에 저장.

017 : E59F9E98;

1. 1110 0101 1001 1111 1001 1110 1001 1000
2. LDR $9, [$15, #0xE98]
3. I bit = 0 이기 때문에 immediate value 를 의미

4. 레지스터 9 에 든 데이터를 레지스터 15 에 든 값과 상수값 #0x E98 을 더한 메모리 주소에 저장.

018 : E3A08000;

1. 1110 0011 1010 0000 1000 0000 0000 0000
2. MOV $8, #0x000
3. I bit = 1 이기 때문에 이는 immediate value 를 의미
4. 레지스터 8 에 상수 값 #0x000 을 저장한다.

019 : E5898000;

1. 1110 0101 1000 1001 1000 0000 0000 0000
2. STR $8, [$9, #0x000]
3. I bit = 0 이기 때문에 immediate value 를 의미
4. 레지스터 8 에 든 데이터를 레지스터 9 에 든 값과 상수값 #0x000 을 더한 메모리 주소에 저장.

01A : E5898004;

1. 1110 0101 1000 1001 1000 0000 0000 0100
2. STR $8, [$9, #0x004]
3. I bit = 0 이기 때문에 immediate value 를 의미
4. 레지스터 8 에 든 데이터를 레지스터 9 에 든 값과 상수값 #0x004 을 더한 메모리 주소에 저장.

01B : E5898008;

1. 1110 0101 1000 1001 1000 0000 0000 1000
2. STR $8, [$9, #0x008]
3. I bit = 0 이기 때문에 immediate value 를 의미
4. 레지스터 8 에 든 데이터를 레지스터 9 에 든 값과 상수값 #0x008 을 더한 메모리 주소에 저장.

01C : E589800C;

1. 1110 0101 1000 1001 1000 0000 0000 1100
2. STR $8, [$9, #0x00C]
3. I bit = 0 이기 때문에 immediate value 를 의미
4. 레지스터 8 에 든 데이터를 레지스터 9 에 든 값과 상수값 #0x00C 을 더한 메모리 주소에 저장.

01D : E5898010;

1. 1110 0101 1000 1001 1000 0000 0001 0000

2. STR $8, [$9, #0x010]
3. I bit = 0 이기 때문에 immediate value 를 의미
4. 레지스터 8 에 든 데이터를 레지스터 9 에 든 값과 상수값 #0x010 을 더한 메모리 주소에 저장.

01E : E5898014;

1. 1110 0101 1000 1001 1000 0000 0001 0100
2. STR $8, [$9, #0x014]
3. I bit = 0 이기 때문에 immediate value 를 의미
4. 레지스터 8 에 든 데이터를 레지스터 9 에 든 값과 상수값 #0x014 을 더한 메모리 주소에 저장.

01F : E5898018;

1. 1110 0101 1000 1001 1000 0000 0001 1000
2. STR $8, [$9, #0x018]
3. I bit = 0 이기 때문에 immediate value 를 의미
4. 레지스터 8 에 든 데이터를 레지스터 9 에 든 값과 상수값 #0x018 을 더한 메모리 주소에 저장.

020 : E59FDE78;

1. 1110 0101 1001 1111 1101 0111 1000
2. LDR $13, [$15, #0xE78]
3. I bit = 0 이기 때문에 immediate value 를 의미
4. 레지스터 13 에 든 데이터를 레지스터 15 에 든 값과 상수값 #0x E78 을 더한 메모리 주소에 저장.

021 : E5931200;

1. 1110 0101 1001 0011 0001 0010 0000 0000
2. LDR $1, [$3, #0x200]
3. I bit = 0 이기 때문에 immediate value 를 의미
4. 레지스터 1 에 든 데이터를 레지스터 3 에 든 값과 상수값 #0x 200 을 더한 메모리 주소에 저장.

022 : E3510001;

1. 1110 0011 0101 0001 0000 0000 0000 0001
2. CMP $1, #0x1
3. 레지스터 1 의 값과 immediate value #1 을 compare 한다.
4. Compare 과정에서 만약에 equal 하다면, flag Z 가 set (=value 1)이 된다.
5. Equal 이 아니라면, flag Z 는 reset (= value 0)가 된다.

023 : 0A000000;

1. 0000 1010 0000 0000 0000 0000 0000 0000
2. BEQ #0
3. 31- 28 bit 가 0000 이기에 EQ
4. 022 주소에서 equal 해서 flag 가 set 이라면, PC + 8 + 4 * 0 = PC + 8, 025 주소로 넘어가 실행.
5. 022 주소가 unequal 해서 flag 가 reset 이라면, PC + 4, 024 주소로 넘어가 실행한다.

024 : EAFFFFFB;

1. 1110 1010 1111 1111 1111 1111 1111 1011
2. B #(-5)
3. Unconditional branch 이므로 PC + 8 + 4*(-5) = PC + 4 * (-3), 021 주소로 넘어가 실행한다.