과목: 컴퓨터구조

교수: 김영근 교수님

컴퓨터구조 <과제 1>

- ARM Instructions 분석 -

제출일자: 2021 / 04 / 16

이름: 홍지훈

학과: 소프트웨어학부

분반: 나

학번 : 20201777

0. insts_data.mif

```
/* TODO */
        000   :   EA000006;
        001   :   EAFFFFFE;
        002   :   EA0000A7;
        [003..005]   :   EAFFFFFE;
        006   :   EA0000A4;
        007   :   EAFFFFFE;
        008   :   E59F2EC8;
        009   :   E3A00040;
        00A   :   E5820010;
        00B   :   E5820014;
        00C   :   E5820018;
        00D   :   E582001C;
        00E   :   E5820020;
        00F   :   E5820024;
        010   :   E3A0003F;
        011   :   E5820028;
        012   :   E3A00008;
        013   :   E582002C;
        014   :   E59F3E9C;
        015   :   E59F1E9C;
        016   :   E5831000;
        017   :   E59F9E98;
        018   :   E3A08000;
        019   :   E5898000;
        01A   :   E5898004;
        01B   :   E5898008;
        01C   :   E589800C;
        01D   :   E5898010;
        01E   :   E5898014;
        01F   :   E5898018;
        020   :   E59FDE78;
        021   :   E5931200;
        022   :   E3510001;
        023   :   0A000000;
        024   :   EAFFFFFB;
```

000 : EA000006

Instruction을 Binary로 변환

1110 1010 0000 0000 0000 0000 0000 0110

어떤 Instruction인지 Reference File을 통해 확인

Branch and branch with link (B, BL) (160P)

- **B #6;**

Instruction이 어떤 의미를 가지는지 서술

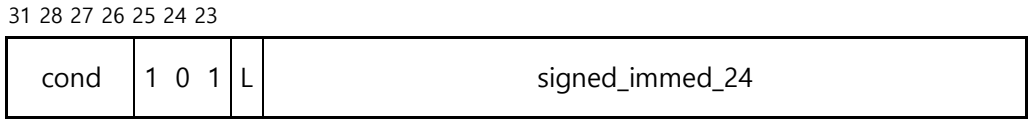
PC+8+6*4 주소로 이동

PC: 현재 실행중인 명령어의 주소

주소 단위는 4Byte (= 1 Word)

따라서 (0 + 8 + 24) / 4 = 008번 주소로 이동

008 : E59F2EC8; 로 이동



001 : EAFFFFFE;

Instruction을 Binary로 변환

1110 1010 1111 1111 1111 1111 1111 1110

어떤 Instruction인지 Reference File을 통해 확인

Branch and branch with link (B, BL) (160P)

- **B #-2;**

Instruction이 어떤 의미를 가지는지 서술

2의 보수를 사용하여 값을 구함

(1111 1111 1111 1111 1111 1110

->0000 0000 0000 0000 0000 0010)(-2)

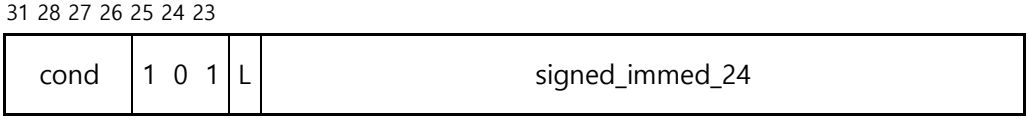
PC+8+(-2)*4 주소로 이동

PC: 현재 실행중인 명령어의 주소

주소 단위는 4Byte (= 1 Word)

따라서 (4 + 8 - 8) / 4 = 001번 주소로 이동

001 : EAFFFFFE; 로 이동 (무한 루프)



002 : EA0000A7;

Instruction을 Binary로 변환

1110 1010 0000 0000 0000 0000 1010 0111

어떤 Instruction인지 Reference File을 통해 확인

Branch and branch with link (B, BL) (160P)

- **B #0x0000A7; (167)**

Instruction이 어떤 의미를 가지는지 서술

2의 보수를 사용하여 값을 구함

(1111 1111 1111 1111 1111 1110

->0000 0000 0000 0000 0000 0010)(-2)

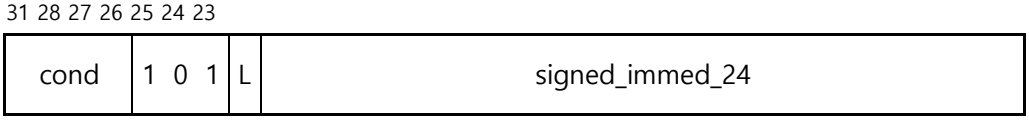
PC+8+167*4 주소로 이동

PC: 현재 실행중인 명령어의 주소

주소 단위는 4Byte (= 1 Word)

따라서 (8 + 8 + 668) / 4 = 171번 주소(hex: 0AB)로 이동

0AB : ????????; 로 이동



003 : EAFFFFFE;

Instruction을 Binary로 변환

1110 1010 1111 1111 1111 1111 1111 1110

어떤 Instruction인지 Reference File을 통해 확인

Branch and branch with link (B, BL) (160P)

- **B #-2;**

Instruction이 어떤 의미를 가지는지 서술

2의 보수를 사용하여 값을 구함

(1111 1111 1111 1111 1111 1110

->0000 0000 0000 0000 0000 0010)(-2)

PC+8+(-2)*4 주소로 이동

PC: 현재 실행중인 명령어의 주소

주소 단위는 4Byte (= 1 Word)

따라서 (12 + 8 - 8) / 4 = 003번 주소로 이동

003 : EAFFFFFE; 로 이동 (무한 루프)

31 28 27 26 25 24 23

cond	1 0 1	L	signed_immed_24

004 : EAFFFFFE;

Instruction을 Binary로 변환

1110 1010 1111 1111 1111 1111 1111 1110

어떤 Instruction인지 Reference File을 통해 확인

Branch and branch with link (B, BL) (160P)

- **B #-2;**

Instruction이 어떤 의미를 가지는지 서술

2의 보수를 사용하여 값을 구함

(1111 1111 1111 1111 1111 1110

->0000 0000 0000 0000 0000 0010)(-2)

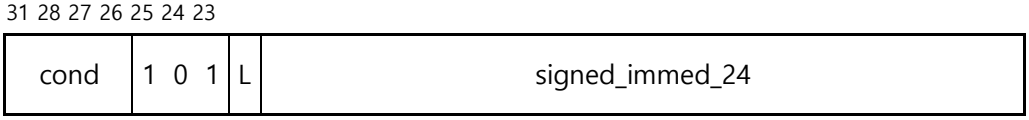
PC+8+(-2)*4 주소로 이동

PC: 현재 실행중인 명령어의 주소

주소 단위는 4Byte (= 1 Word)

따라서 (16 + 8 - 8) / 4 = 004번 주소로 이동

004 : EAFFFFFE; 로 이동 (무한 루프)



005 : EAFFFFFE;

Instruction을 Binary로 변환

1110 1010 1111 1111 1111 1111 1111 1110

어떤 Instruction인지 Reference File을 통해 확인

Branch and branch with link (B, BL) (160P)

- **B #-2;**

Instruction이 어떤 의미를 가지는지 서술

2의 보수를 사용하여 값을 구함

(1111 1111 1111 1111 1111 1110

->0000 0000 0000 0000 0000 0010)(-2)

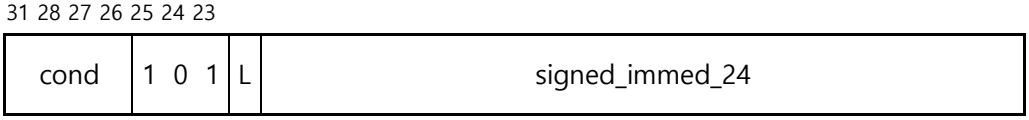
PC+8+(-2)*4 주소로 이동

PC: 현재 실행중인 명령어의 주소

주소 단위는 4Byte (= 1 Word)

따라서 (20 + 8 - 8) / 4 = 005번 주소로 이동

005 : EAFFFFFE; 로 이동 (무한 루프)



006 : EA0000A4;

Instruction을 Binary로 변환

1110 1010 0000 0000 0000 0000 1010 0100

어떤 Instruction인지 Reference File을 통해 확인

Branch and branch with link (B, BL) (160P)

- **B #0x0000A4; (164)**

Instruction이 어떤 의미를 가지는지 서술

2의 보수를 사용하여 값을 구함

(1111 1111 1111 1111 1111 1110

->0000 0000 0000 0000 0000 0010)(-2)

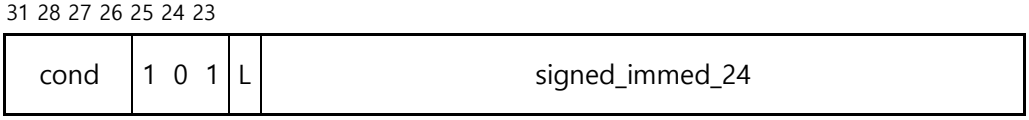
PC+8+164*4 주소로 이동

PC: 현재 실행중인 명령어의 주소

주소 단위는 4Byte (= 1 Word)

따라서 (24 + 8 + 656) / 4 = 172번 주소로 이동

172 : ????????; 로 이동



007 : EAFFFFFE;

Instruction을 Binary로 변환

1110 1010 1111 1111 1111 1111 1111 1110

어떤 Instruction인지 Reference File을 통해 확인

Branch and branch with link (B, BL) (160P)

- **B #-2;**

Instruction이 어떤 의미를 가지는지 서술

2의 보수를 사용하여 값을 구함

(1111 1111 1111 1111 1111 1110

->0000 0000 0000 0000 0000 0010)(-2)

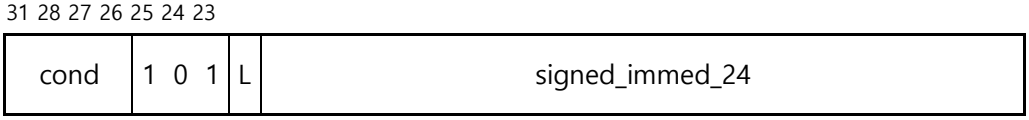
PC+8+(-2)*4 주소로 이동

PC: 현재 실행중인 명령어의 주소

주소 단위는 4Byte (= 1 Word)

따라서 (28 + 8 - 8) / 4 = 007번 주소로 이동

007 : EAFFFFFE; 로 이동 (무한 루프)



008 : E59F2EC8;

Instruction을 Binary로 변환

1110 0101 1001 1111 0010 1110 1100 1000

어떤 Instruction인지 Reference File을 통해 확인

Load/store immediate offset (LDR, STR) (129P)

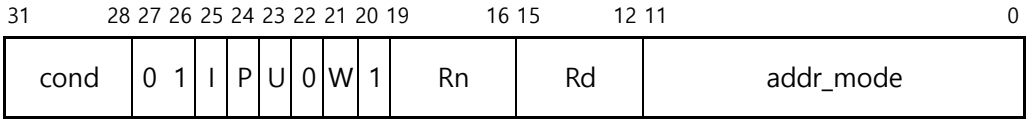
- **LDR $2, [$15, #0xEC8];**

Instruction이 어떤 의미를 가지는지 서술

15번 레지스터에 저장된 값에 #0xEC8값을 더한 주소의 값을 메모리로부터 읽어와서 2번 레지스터에 저장

메모리의 [$15 + #0xEC8] 주소에 저장된 값을 읽어와 $2에 저장

A4.1.23 LDR



009 : E3A00040;

Instruction을 Binary로 변환

1110 **0011** 1010 0000 0000 0000 0100 0000

어떤 Instruction인지 Reference File을 통해 확인

Data processing immediate [2] (MOV) (115P)

- **MOV $0, #0x40;**

Instruction이 어떤 의미를 가지는지 서술

Opcode: 1101 = MOV

0번 레지스터에 #0x40의 값을 저장

A4.1.35 MOV

31

cond	0 0	I	1 1 0 1	S	SBZ	Rd	shifter_operand

00A : E5820010;

Instruction을 Binary로 변환

1110 **010**1 1000 0010 0000 0000 0001 0000

어떤 Instruction인지 Reference File을 통해 확인

Load/store immediate offset (LDR, STR) (129P)

- **STR $0, [$2, #0x010];**

Instruction이 어떤 의미를 가지는지 서술

2번 레지스터에 저장된 값에 #0x010값을 더해 주소 값을 계산

0번 레지스터에 저장되어 있던 값을 위에서 계산한 메모리 주소에 저장

A4.1.99 STR

31

cond	0 1	I	P	U	0	W	0	Rn	Rd	addr_mode

STR (Store Register) stores a word from a register to memory.

00B : E5820014;

Instruction을 Binary로 변환

1110 **010**1 1000 0010 0000 0000 0001 0100

어떤 Instruction인지 Reference File을 통해 확인

Load/store immediate offset (LDR, STR) (129P)

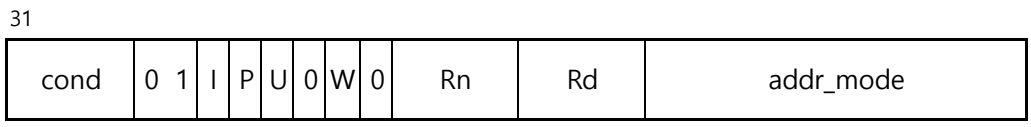
- **STR $0, [$2, #0x014];**

Instruction이 어떤 의미를 가지는지 서술

2번 레지스터에 저장된 값에 #0x014값을 더해 주소 값을 계산

0번 레지스터에 저장되어 있던 값을 위에서 계산한 메모리 주소에 저장

A4.1.99 STR



STR (Store Register) stores a word from a register to memory.

00C : E5820018;

Instruction을 Binary로 변환

1110 **010**1 1000 0010 0000 0000 0001 1000

어떤 Instruction인지 Reference File을 통해 확인

Load/store immediate offset (LDR, STR) (129P)

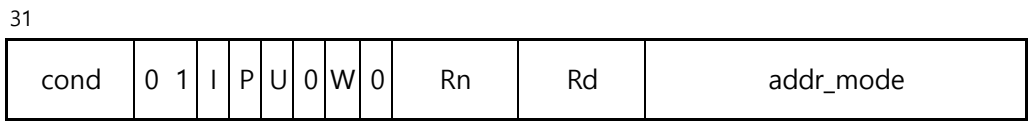
- **STR $0, [$2, #0x018];**

Instruction이 어떤 의미를 가지는지 서술

2번 레지스터에 저장된 값에 #0x018값을 더해 주소 값을 계산

0번 레지스터에 저장되어 있던 값을 위에서 계산한 메모리 주소에 저장

A4.1.99 STR



STR (Store Register) stores a word from a register to memory.

00D : E582001C;

Instruction을 Binary로 변환

1110 **010**1 1000 0010 0000 0000 0001 1100

어떤 Instruction인지 Reference File을 통해 확인

Load/store immediate offset (LDR, STR) (129P)

- **STR $0, [$2, #0x01C];**

Instruction이 어떤 의미를 가지는지 서술

2번 레지스터에 저장된 값에 #0x01C값을 더해 주소 값을 계산

0번 레지스터에 저장되어 있던 값을 위에서 계산한 메모리 주소에 저장

A4.1.99 STR

31

cond	0 1	I	P	U	0	W	0	Rn	Rd	addr_mode

STR (Store Register) stores a word from a register to memory.

00E : E5820020;

Instruction을 Binary로 변환

1110 **010**1 1000 0010 0000 0000 0010 0000

어떤 Instruction인지 Reference File을 통해 확인

Load/store immediate offset (LDR, STR) (129P)

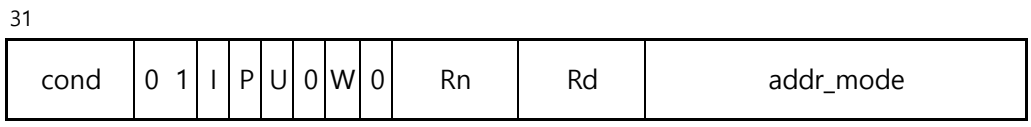
- **STR $0, [$2, #0x020];**

Instruction이 어떤 의미를 가지는지 서술

2번 레지스터에 저장된 값에 #0x020값을 더해 주소 값을 계산

0번 레지스터에 저장되어 있던 값을 위에서 계산한 메모리 주소에 저장

A4.1.99 STR



STR (Store Register) stores a word from a register to memory.

00F : E5820024;

Instruction을 Binary로 변환

1110 **010**1 1000 0010 0000 0000 0010 0100

어떤 Instruction인지 Reference File을 통해 확인

Load/store immediate offset (LDR, STR) (129P)

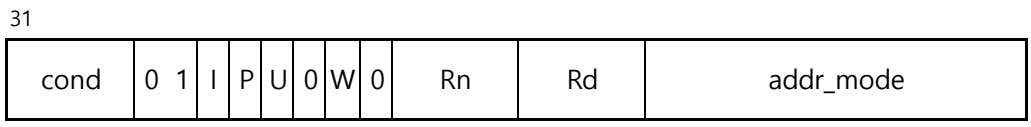
- **STR $0, [$2, #0x024];**

Instruction이 어떤 의미를 가지는지 서술

2번 레지스터에 저장된 값에 #0x024값을 더해 주소 값을 계산

0번 레지스터에 저장되어 있던 값을 위에서 계산한 메모리 주소에 저장

A4.1.99 STR



STR (Store Register) stores a word from a register to memory.

010 : E3A0003F;

Instruction을 Binary로 변환

1110 **0011** 1010 0000 0000 0000 0011 1111

어떤 Instruction인지 Reference File을 통해 확인

Data processing immediate [2] (MOV) (115P)

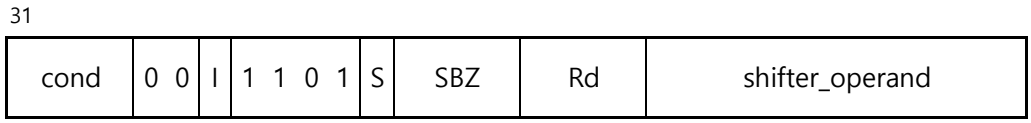
- **MOV $0, #0x03F;**

Instruction이 어떤 의미를 가지는지 서술

Opcode: 1101 = MOV

0번 레지스터에 #0x03F의 값을 저장

A4.1.35 MOV



011 : E5820028;

Instruction을 Binary로 변환

1110 **010**1 1000 0010 0000 0000 0010 1000

어떤 Instruction인지 Reference File을 통해 확인

Load/store immediate offset (LDR, STR) (129P)

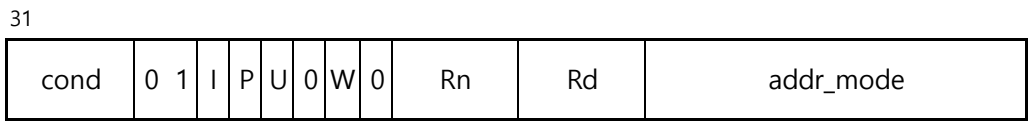
- **STR $0, [$2, #0x028];**

Instruction이 어떤 의미를 가지는지 서술

2번 레지스터에 저장된 값에 #0x028값을 더해 주소 값을 계산

0번 레지스터에 저장되어 있던 값을 위에서 계산한 메모리 주소에 저장

A4.1.99 STR



STR (Store Register) stores a word from a register to memory.

012 : E3A00008;

Instruction을 Binary로 변환

1110 **0011** 1010 0000 0000 0000 0000 1000

어떤 Instruction인지 Reference File을 통해 확인

Data processing immediate [2] (MOV) (115P)

- **MOV $0, #0x008;**

Instruction이 어떤 의미를 가지는지 서술

Opcode: 1101 = MOV

0번 레지스터에 #0x008의 값을 저장

A4.1.35 MOV

31

cond	0 0	I	1 1 0 1	S	SBZ	Rd	shifter_operand

013 : E582002C;

Instruction을 Binary로 변환

1110 **010**1 1000 0010 0000 0000 0010 1100

어떤 Instruction인지 Reference File을 통해 확인

Load/store immediate offset (LDR, STR) (129P)

- **STR $0, [$2, #0x02C];**

Instruction이 어떤 의미를 가지는지 서술

2번 레지스터에 저장된 값에 #0x02C값을 더해 주소 값을 계산

0번 레지스터에 저장되어 있던 값을 위에서 계산한 메모리 주소에 저장

A4.1.99 STR

31

cond	0 1	I	P	U	0	W	0	Rn	Rd	addr_mode

STR (Store Register) stores a word from a register to memory.

014 : E59F3E9C;

Instruction을 Binary로 변환

1110 0101 1001 1111 0011 1110 1001 1100

어떤 Instruction인지 Reference File을 통해 확인

Load/store immediate offset (LDR, STR) (129P)

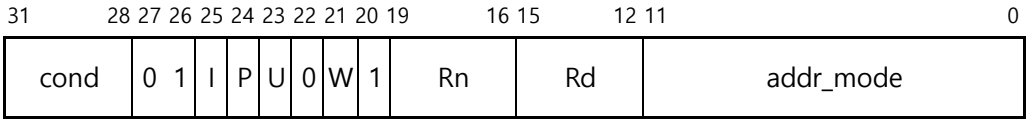
- **LDR $3, [$15, #0xE9C];**

Instruction이 어떤 의미를 가지는지 서술

15번 레지스터에 저장된 값에 #0xE9C값을 더한 주소의 값을 메모리로부터 읽어와서 3번 레지스터에 저장

메모리의 [$15 + #0xE9C] 주소에 저장된 값을 읽어와 $3에 저장

A4.1.23 LDR



015 : E59F1E9C;

Instruction을 Binary로 변환

1110 0101 1001 1111 0001 1110 1001 1100

어떤 Instruction인지 Reference File을 통해 확인

Load/store immediate offset (LDR, STR) (129P)

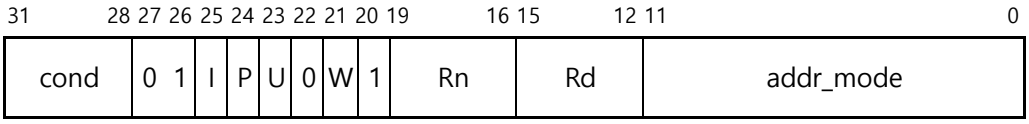
- **LDR $1, [$15, #0xE9C];**

Instruction이 어떤 의미를 가지는지 서술

15번 레지스터에 저장된 값에 #0xE9C값을 더한 주소의 값을 메모리로부터 읽어와서 1번 레지스터에 저장

메모리의 [$15 + #0xE9C] 주소에 저장된 값을 읽어와 $1에 저장

A4.1.23 LDR



016 : E5831000;

Instruction을 Binary로 변환

1110 **010**1 1000 0011 0001 0000 0000 0000

어떤 Instruction인지 Reference File을 통해 확인

Load/store immediate offset (LDR, STR) (129P)

- **STR $1, [$3, #0x000];**

Instruction이 어떤 의미를 가지는지 서술

3번 레지스터에 저장된 값에 #0x000값을 더해 주소 값을 계산

1번 레지스터에 저장되어 있던 값을 위에서 계산한 메모리 주소에 저장

A4.1.99 STR

31

cond	0 1	I	P	U	0	W	0	Rn	Rd	addr_mode

STR (Store Register) stores a word from a register to memory.

017 : E59F9E98;

Instruction을 Binary로 변환

1110 0101 1001 1111 1001 1110 1001 1000

어떤 Instruction인지 Reference File을 통해 확인

Load/store immediate offset (LDR, STR) (129P)

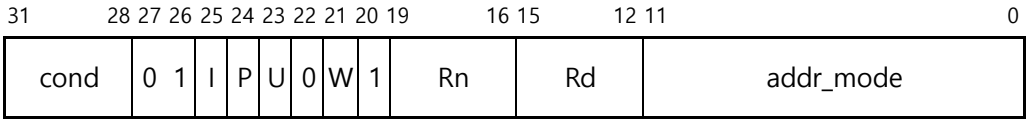
- **LDR $9, [$15, #0xE98];**

Instruction이 어떤 의미를 가지는지 서술

15번 레지스터에 저장된 값에 #0xE98값을 더한 주소의 값을 메모리로부터 읽어와서 9번 레지스터에 저장

메모리의 [$15 + #0xE98] 주소에 저장된 값을 읽어와 $9에 저장

A4.1.23 LDR



018 : E3A08000;

Instruction을 Binary로 변환

1110 **0011** 1010 0000 1000 0000 0000 0000

어떤 Instruction인지 Reference File을 통해 확인

Data processing immediate [2] (MOV) (115P)

- **MOV $8, #0x000;**

Instruction이 어떤 의미를 가지는지 서술

Opcode: 1101 = MOV

8번 레지스터에 #0x000의 값을 저장

A4.1.35 MOV

31

cond	0 0	I	1 1 0 1	S	SBZ	Rd	shifter_operand

019 : E5898000;

Instruction을 Binary로 변환

1110 **010**1 1000 1001 1000 0000 0000 0000

어떤 Instruction인지 Reference File을 통해 확인

Load/store immediate offset (LDR, STR) (129P)

- **STR $8, [$9, #0x000];**

Instruction이 어떤 의미를 가지는지 서술

9번 레지스터에 저장된 값에 #0x000값을 더해 주소 값을 계산

8번 레지스터에 저장되어 있던 값을 위에서 계산한 메모리 주소에 저장

A4.1.99 STR

31

cond	0 1	I	P	U	0	W	0	Rn	Rd	addr_mode

STR (Store Register) stores a word from a register to memory.

01A : E5898004;

Instruction을 Binary로 변환

1110 **010**1 1000 1001 1000 0000 0000 0100

어떤 Instruction인지 Reference File을 통해 확인

Load/store immediate offset (LDR, STR) (129P)

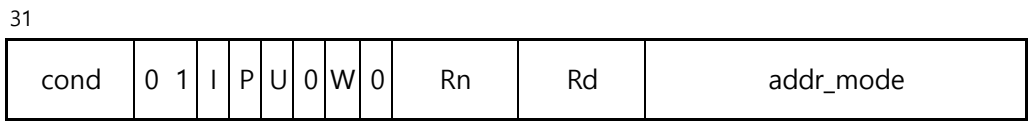
- **STR $8, [$9, #0x004];**

Instruction이 어떤 의미를 가지는지 서술

9번 레지스터에 저장된 값에 #0x004값을 더해 주소 값을 계산

8번 레지스터에 저장되어 있던 값을 위에서 계산한 메모리 주소에 저장

A4.1.99 STR



STR (Store Register) stores a word from a register to memory.

01B : E5898008;

Instruction을 Binary로 변환

1110 **010**1 1000 1001 1000 0000 0000 1000

어떤 Instruction인지 Reference File을 통해 확인

Load/store immediate offset (LDR, STR) (129P)

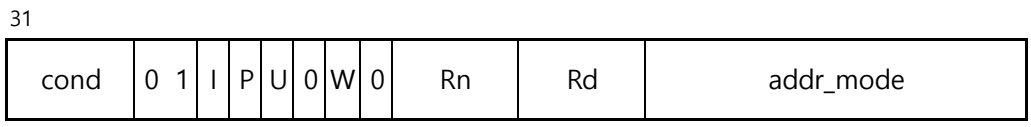
- **STR $8, [$9, #0x008];**

Instruction이 어떤 의미를 가지는지 서술

9번 레지스터에 저장된 값에 #0x008값을 더해 주소 값을 계산

8번 레지스터에 저장되어 있던 값을 위에서 계산한 메모리 주소에 저장

A4.1.99 STR



STR (Store Register) stores a word from a register to memory.

01C : E589800C;

Instruction을 Binary로 변환

1110 **010**1 1000 1001 1000 0000 0000 1100

어떤 Instruction인지 Reference File을 통해 확인

Load/store immediate offset (LDR, STR) (129P)

- **STR $8, [$9, #0x00C];**

Instruction이 어떤 의미를 가지는지 서술

9번 레지스터에 저장된 값에 #0x00C값을 더해 주소 값을 계산

8번 레지스터에 저장되어 있던 값을 위에서 계산한 메모리 주소에 저장

A4.1.99 STR

31

cond	0 1	I	P	U	0	W	0	Rn	Rd	addr_mode

STR (Store Register) stores a word from a register to memory.

01D : E5898010;

Instruction을 Binary로 변환

1110 **010**1 1000 1001 1000 0000 0001 0000

어떤 Instruction인지 Reference File을 통해 확인

Load/store immediate offset (LDR, STR) (129P)

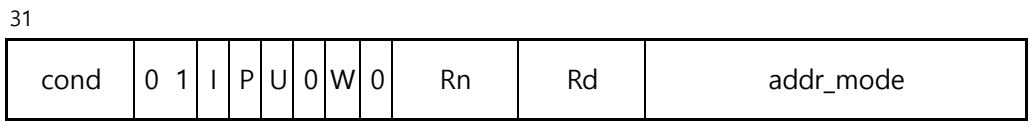
- **STR $8, [$9, #0x010];**

Instruction이 어떤 의미를 가지는지 서술

9번 레지스터에 저장된 값에 #0x010값을 더해 주소 값을 계산

8번 레지스터에 저장되어 있던 값을 위에서 계산한 메모리 주소에 저장

A4.1.99 STR



STR (Store Register) stores a word from a register to memory.

01E : E5898014;

Instruction을 Binary로 변환

1110 **010**1 1000 1001 1000 0000 0001 0100

어떤 Instruction인지 Reference File을 통해 확인

Load/store immediate offset (LDR, STR) (129P)

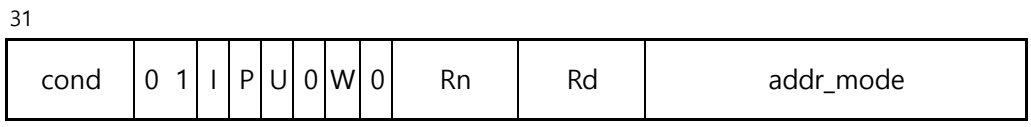
- **STR $8, [$9, #0x014];**

Instruction이 어떤 의미를 가지는지 서술

9번 레지스터에 저장된 값에 #0x014값을 더해 주소 값을 계산

8번 레지스터에 저장되어 있던 값을 위에서 계산한 메모리 주소에 저장

A4.1.99 STR



STR (Store Register) stores a word from a register to memory.

01F : E5898018;

Instruction을 Binary로 변환

1110 **010**1 1000 1001 1000 0000 0001 1000

어떤 Instruction인지 Reference File을 통해 확인

Load/store immediate offset (LDR, STR) (129P)

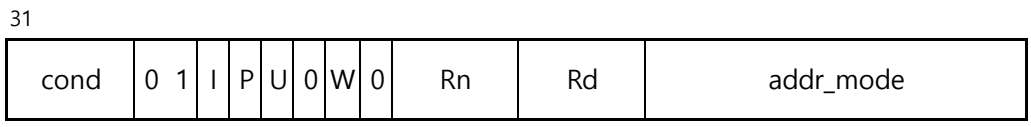
- **STR $8, [$9, #0x018];**

Instruction이 어떤 의미를 가지는지 서술

9번 레지스터에 저장된 값에 #0x018값을 더해 주소 값을 계산

8번 레지스터에 저장되어 있던 값을 위에서 계산한 메모리 주소에 저장

A4.1.99 STR



STR (Store Register) stores a word from a register to memory.

020 : E59FDE78;

Instruction을 Binary로 변환

1110 0101 1001 1111 1101 1110 0111 1000

어떤 Instruction인지 Reference File을 통해 확인

Load/store immediate offset (LDR, STR) (129P)

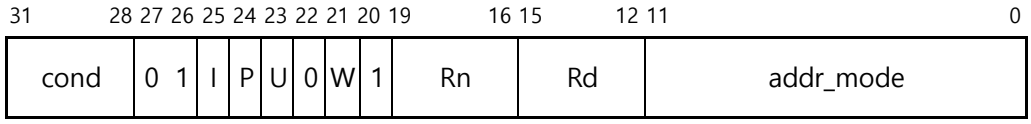
- **LDR $13, [$15, #0xE78];**

Instruction이 어떤 의미를 가지는지 서술

15번 레지스터에 저장된 값에 #0xE78값을 더한 주소의 값을 메모리로부터 읽어와서 13번 레지스터에 저장

메모리의 [$15 + #0xE78] 주소에 저장된 값을 읽어와 $13에 저장

A4.1.23 LDR



021 : E5931200;

Instruction을 Binary로 변환

1110 0101 1001 0011 0001 0010 0000 0000

어떤 Instruction인지 Reference File을 통해 확인

Load/store immediate offset (LDR, STR) (129P)

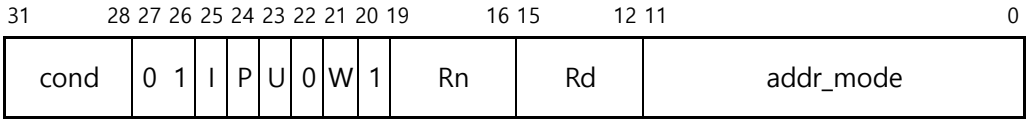
- **LDR $1, [$3, #0x200];**

Instruction이 어떤 의미를 가지는지 서술

3번 레지스터에 저장된 값에 #0x200값을 더한 주소의 값을 메모리로부터 읽어와서 1번 레지스터에 저장

메모리의 [$3 + #0x200] 주소에 저장된 값을 읽어와 $1에 저장

A4.1.23 LDR



022 : E3510001;

Instruction을 Binary로 변환

1110 **0011** 0101 0001 0000 0000 0000 0001

어떤 Instruction인지 Reference File을 통해 확인

Data processing immediate [2] (CMP) (115P)

- **CMP $1, #0x001;**

Instruction이 어떤 의미를 가지는지 서술

Opcode: 1010 = CMP

if ConditionPassed(cond) then

alu_out = Rn - shifter_operand

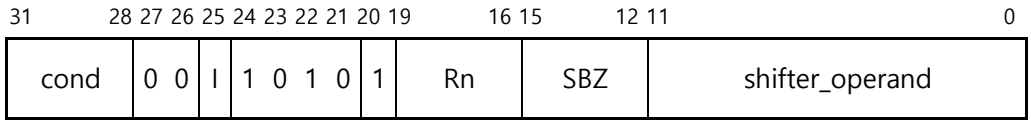
Z Flag = if alu_out == 0 then 1 else 0

1번 레지스터에서의 값과 #0x008의 값을 비교

#0x008에서 1번 레지스터의 값을 빼서

그 값이 0이면 Z Flag = 1, 아니면 Z Flag = 0이 된다.

A4.1.15 CMP



CMP (Compare) compares two values. The first value comes from a register. The second value can be either an immediate value or a value from a register, and can be shifted before the comparison.

CMP updates the condition flags, based on the result of subtracting the second value from the first.

023 : 0A000000;

Instruction을 Binary로 변환

0000 **101**0 0000 0000 0000 0000 0000 0000

어떤 Instruction인지 Reference File을 통해 확인

Branch and branch with link (B, BL) (160P)

- **BEQ #0;**

Instruction이 어떤 의미를 가지는지 서술

Z Flag가 set되어 있으면

PC+8+0*4 주소로 이동

따라서 (140 + 8 + 0) / 4 = 037번 주소(hex: 025)로 이동

025 : ????????; 로 이동

A4.1.99 STR

31

cond	0 1	I	P	U	0	W	0	Rn	Rd	addr_mode

STR (Store Register) stores a word from a register to memory.

024 : EAFFFFFB;

Instruction을 Binary로 변환

1110 1010 1111 1111 1111 1111 1111 1011

어떤 Instruction인지 Reference File을 통해 확인

Branch and branch with link (B, BL) (160P)

- **B #-2;**

Instruction이 어떤 의미를 가지는지 서술

2의 보수를 사용하여 값을 구함

(1111 1111 1111 1111 1111 1011

->0000 0000 0000 0000 0000 0101)(-5)

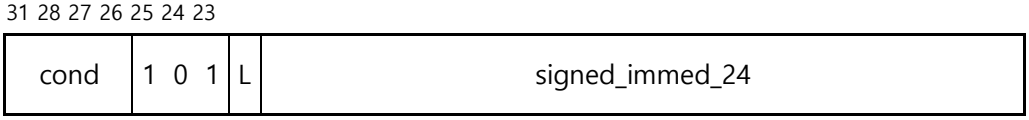
PC+8+(-2)*4 주소로 이동

PC: 현재 실행중인 명령어의 주소

주소 단위는 4Byte (= 1 Word)

따라서 (144 + 8 - 20) / 4 = 033번 주소(hex:021)로 이동

021 : E5931200; 로 이동



동작과정

000 : EA000006; -> 008 : E59F2EC8;로 이동

008 : E59F2EC8; -> 2번 레지스터에 메모리의 [15번 레지스터 + #0xEC8] 주소에 저장된 값을 저장

009 : E3A00040; -> 0번 레지스터에 #0x040값을 저장

00A : E5820010; -> 2번 레지스터에 저장된 값에 #0x010값을 더한 주소 값에 0번 레지스터에 저장되어 있떤 값을 저장

00B : E5820014; -> 2번 레지스터에 저장된 값에 #0x014값을 더한 주소 값에 0번 레지스터에 저장되어 있던 값을 저장

00C : E5820018; -> 2번 레지스터에 저장된 값에 #0x018값을 더한 주소 값에 0번 레지스터에 저장되어 있던 값을 저장

00D : E582001C; -> 2번 레지스터에 저장된 값에 #0x01C값을 더한 주소 값에 0번 레지스터에 저장되어 있던 값을 저장

00E : E5820020; -> 2번 레지스터에 저장된 값에 #0x020값을 더한 주소 값에 0번 레지스터에 저장되어 있던 값을 저장

00F : E5820024; -> 2번 레지스터에 저장된 값에 #0x024값을 더한 주소 값에 0번 레지스터에 저장되어 있던 값을 저장

010 : E3A0003F; -> 0번 레지스터에 #0x03F값을 저장

011 : E5820028; -> 2번 레지스터에 저장된 값에 #0x028값을 더한 주소 값에 0번 레지스터에 저장되어 있던 값을 저장

012 : E3A00008; -> 0번 레지스터에 #0x008값을 저장

013 : E582002C; -> 2번 레지스터에 저장된 값에 #0x02C값을 더한 주소 값에 0번 레지스터에 저장되어 있던 값을 저장

014 : E59F3E9C; -> 3번 레지스터에 메모리의 [15번 레지스터 + #0xE9C] 주소에 저장된 값을 저장

015 : E59F1E9C; -> 1번 레지스터에 메모리의 [15번 레지스터 + #0xE9C] 주소에 저장된 값을 저장

016 : E5831000; -> 3번 레지스터에 저장된 값에 #0x000값을 더한 주소 값에 1번 레지스터에 저장되어 있던 값을 저장

017 : E59F9E98; -> 9번 레지스터에 메모리의 [15번 레지스터 + #0xE98] 주소에 저장된 값을 저장

018 : E3A08000; -> 8번 레지스터에 #0x000값을 저장

019 : E5898000; -> 9번 레지스터에 저장된 값에 #0x000값을 더한 주소 값에 8번 레지스터에 저장되어 있던 값을 저장
01A : E5898004; -> 9번 레지스터에 저장된 값에 #0x004값을 더한 주소 값에 8번 레지스터에 저장되어 있던 값을 저장
01B : E5898008; -> 9번 레지스터에 저장된 값에 #0x008값을 더한 주소 값에 8번 레지스터에 저장되어 있던 값을 저장
01C : E589800C; -> 9번 레지스터에 저장된 값에 #0x00C값을 더한 주소 값에 8번 레지스터에 저장되어 있던 값을 저장
01D : E5898010; -> 9번 레지스터에 저장된 값에 #0x010값을 더한 주소 값에 8번 레지스터에 저장되어 있던 값을 저장
01E : E5898014; -> 9번 레지스터에 저장된 값에 #0x014값을 더한 주소 값에 8번 레지스터에 저장되어 있던 값을 저장
01F : E5898018; -> 9번 레지스터에 저장된 값에 #0x018값을 더한 주소 값에 8번 레지스터에 저장되어 있던 값을 저장
020 : E59FDE78; -> 13번 레지스터에 메모리의 [15번 레지스터 + #0xE78] 주소에 저장된 값을 저장
021 : E5931200; -> 1번 레지스터에 메모리의 [3번 레지스터 + #0x200] 주소에 저장된 값을 저장
022 : E3510001; -> 1번 레지스터의 값과 #0x008의 값을 비교 후 #0x008에서 1번 레지스터의 값을 뺌.
그 결과값이 0이면 : Z Flag = 1;
아니면 : Z Flag = 0;

023 : 0A000000; ->
Z Flag가 set(참)되어 있으면
025번 주소로 이동

Z Flag가 0이면
024 : EAFFFFFB; -> 021번 주소로 이동

끝나는 지점

(1)

022 : E3510001;에서 #0X008의 값과 1번 레지스터의 값을 뺀 결과가 0이면 Z Flag = 1;

023 : 0A000000;에서 Z Flag가 1(참)이므로 025 : ???????? 주소로 이동

(2)

022 : E3510001;에서 #0X008의 값과 1번 레지스터의 값을 뺀 결과가 0이 아니면 Z Flag = 0;

023 : 0A000000;에서 Z Flag가 0(거짓)이므로 계속 진행

024 : EAFFFFFB;에서 021주소로 이동 후 022주소의 결과(Z Flag)가 1이 될 때까지 반복