과목: 컴퓨터구조

교수: 김영근 교수님

컴퓨터구조 <과제 1>

- ARM Instructions 분석 -

제출일자: 2021 / 04 /

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**

31 28 27 26 25 24 23

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| cond | 1 | 0 | 1 | L | signed\_immed\_24 |

# 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)**

31 28 27 26 25 24 23

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| cond | 1 | 0 | 1 | L | signed\_immed\_24 |

# 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 주소로 이동

**171 : ????????; 로 이동**

31 28 27 26 25 24 23

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| cond | 1 | 0 | 1 | L | signed\_immed\_24 |

# 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)**

31 28 27 26 25 24 23

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| cond | 1 | 0 | 1 | L | signed\_immed\_24 |

# 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)**

31 28 27 26 25 24 23

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| cond | 1 | 0 | 1 | L | signed\_immed\_24 |

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**

31 28 27 26 25 24 23

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| cond | 1 | 0 | 1 | L | signed\_immed\_24 |

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)**

31 28 27 26 25 24 23

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| cond | 1 | 0 | 1 | L | signed\_immed\_24 |

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**

31 28 27 26 25 24 23 22 21 20 19 16 15 12 11 0

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| cond | 0 | 1 | I | P | U | 0 | W | 1 | Rn | Rd | addr\_mode |

009 : E3A00040;

**Instruction을 Binary로 변환**

1110 **001**1 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**

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.

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**

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.

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**

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.

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**

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.

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**

**Instruction이 어떤 의미를 가지는지 서술**

2번 레지스터에 저장된 값에 #0xEC8값을 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.