보고서
ARM Instructions 분석

소프트웨어학부
20201718
강대겸

Instruction File (inst_data.mif) 내 000번 ~ 024번 주소의 ARM 명령어 분석

- 16진수로 되어 있는 명령어를 2진수 (Binary)로 변환
- 어떤 명령어인지 ARM Reference Manual을 통해 확인
- 해당 명령어가 어떤 의미를 가지는지 서술
- 분석한 의미에 맞게 실제 동작 순서 기록
- 동작이 어디서 끝나는지 명시

(끝나지 않는다면, 어디서부터 어디까지 반복되는지 명시)

000 : EA000006

➢ instruction을 Binary로 변환
1110 1010 0000 0000 0000 0000 0000 0110

➢ 어떤 Instruction인지 Reference File을 통해 확인

B #6

➢ Instruction이 어떤 의미를 가지는지 서술

PC+8+6*4 주소로 이동

(0+8+24)/4 = 008번 주소로 이동

다음 Instruction은 008번 주소에 있는 E59F2EC8

001 : EAFFFFFE

➢ instruction을 Binary로 변환
1110 1010 1111 1111 1111 1111 1111 1110

➢ 어떤 Instruction인지 Reference File을 통해 확인

B #-2

➢ Instruction이 어떤 의미를 가지는지 서술

1. Sign-extending the 24-bit signed immediate to 30 bits
2. Shifting the result left two bits to form a 32-bit value

Adding this to the contents of the PC, which contains the address of the branch instruction plus 8 bytes.

현재 명령어의 주소 (1*4)+8에 위에서 계산한 -8을 더해 4를 만듦

4를 4로 나누어 Word 단위의 명령어 순서 중 1번째로 Branch함

002 : EA0000A7

➢ instruction을 Binary로 변환
1110 1010 0000 0000 0000 0000 1010 0111

➢ 어떤 Instruction인지 Reference File을 통해 확인

B #A7

➢ Instruction이 어떤 의미를 가지는지 서술

PC+8+A7*4 주소로 이동

(8+8+668)/4 = 0AB번 주소로 이동

[003..005] : EAFFFFFE

➢ instruction을 Binary로 변환
1110 1010 1111 1111 1111 1111 1111 1110
➢ 어떤 Instruction인지 Reference File을 통해 확인
B #-2
➢ Instruction이 어떤 의미를 가지는지 서술
1. Sign-extending the 24-bit signed immediate to 30 bits
2. Shifting the result left two bits to form a 32-bit value
Adding this to the contents of the PC, which contains the address of the branch instruction plus 8 bytes.

현재 명령어의 주소 (3~5*4)+8에 위에서 계산한 -8을 더해 12, 16, 20을 만듦
12, 16, 20을 4로 나누어 Word 단위의 명령어 순서 중 3, 4, 5번째로 Branch함

006 : EA0000A4

➢ instruction을 Binary로 변환
1110 1010 0000 0000 0000 0000 1010 0100
➢ 어떤 Instruction인지 Reference File을 통해 확인
B #A4
➢ Instruction이 어떤 의미를 가지는지 서술
PC+8+A7*4 주소로 이동
(24+8+668)/4 = 0A6번 주소로 이동

007 : EAFFFFFE

➢ instruction을 Binary로 변환
1110 1010 1111 1111 1111 1111 1111 1110
➢ 어떤 Instruction인지 Reference File을 통해 확인
B #-2
➢ Instruction이 어떤 의미를 가지는지 서술
1. Sign-extending the 24-bit signed immediate to 30 bits
2. Shifting the result left two bits to form a 32-bit value
Adding this to the contents of the PC, which contains the address of the branch instruction plus 8 bytes.

현재 명령어의 주소 (7*4)+8에 위에서 계산한 -8을 더해 28을 만듦
28을 4로 나누어 Word 단위의 명령어 순서 중 7번째로 Branch함

008 : E59F2EC8

➢ instruction을 Binary로 변환
1110 0101 1001 1111 0010 1110 1100 1000
➢ 어떤 Instruction인지 Reference File을 통해 확인
LDR $2, [$15, #0xEC8];
25번째 bit인 I bit가 0이므로, #으로 표기
➢ Instruction이 어떤 의미를 가지는지 서술
15번 Register에 저장된 값에 #0xEC8값을 더한 주소의 값을 메모리로부터 읽어 와서 2번 Register에 저장
메모리의 [$15 + #0xEC8] 주소에 저장된 값을 읽어와 $2에 저장

009 : E3A00040

➢ instruction을 Binary로 변환
1110 0011 1010 0000 0000 0000 0100 0000

➢ 어떤 Instruction인지 Reference File을 통해 확인

MOV $0, #0x40;

➢ Instruction이 어떤 의미를 가지는지 서술

0번 레지스터에 0x40를 저장

00A : E5820010

➢ instruction을 Binary로 변환
1110 0101 1000 0010 0000 0000 0001 0000

➢ 어떤 Instruction인지 Reference File을 통해 확인

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

25번째 bit인 I bit가 0이므로, #으로 표기

➢ Instruction이 어떤 의미를 가지는지 서술

2번 레지스터에 저장된 값에 #0x010값을 더해 주소 값 계산

0번 레지스터에 저장되어 있던 값을 위에서 계산한 메모리 주소에 저장

00B : E5820014

➢ instruction을 Binary로 변환
1110 0101 1000 0010 0000 0000 0001 0100

➢ 어떤 Instruction인지 Reference File을 통해 확인

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

25번째 bit인 I bit가 0이므로, #으로 표기

➢ Instruction이 어떤 의미를 가지는지 서술

2번 레지스터에 저장된 값에 #0x014값을 더해 주소 값 계산

0번 레지스터에 저장되어 있던 값을 위에서 계산한 메모리 주소에 저장

00C : E5820018

➢ instruction을 Binary로 변환
1110 0101 1000 0010 0000 0000 0001 1000

➢ 어떤 Instruction인지 Reference File을 통해 확인

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

25번째 bit인 I bit가 0이므로, #으로 표기

➢ Instruction이 어떤 의미를 가지는지 서술

2번 레지스터에 저장된 값에 #0x018값을 더해 주소 값 계산

0번 레지스터에 저장되어 있던 값을 위에서 계산한 메모리 주소에 저장

00D : E582001C

➢ instruction을 Binary로 변환
1110 0101 1000 0010 0000 0000 0001 1100

➢ 어떤 Instruction인지 Reference File을 통해 확인

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

25번째 bit인 I bit가 0이므로, #으로 표기

➢ Instruction이 어떤 의미를 가지는지 서술

2번 레지스터에 저장된 값에 #0x01C값을 더해 주소 값 계산
0번 레지스터에 저장되어 있던 값을 위에서 계산한 메모리 주소에 저장

00E : E5820020

➢ instruction을 Binary로 변환
1110 0101 1000 0010 0000 0000 0010 0000
➢ 어떤 Instruction인지 Reference File을 통해 확인
STR $0, [$2, #0x020];
25번째 bit인 I bit가 0이므로, #으로 표기
➢ Instruction이 어떤 의미를 가지는지 서술
2번 레지스터에 저장된 값에 #0x020값을 더해 주소 값 계산
0번 레지스터에 저장되어 있던 값을 위에서 계산한 메모리 주소에 저장

00F : E5820024

➢ instruction을 Binary로 변환
1110 0101 1000 0010 0000 0000 0010 0100
➢ 어떤 Instruction인지 Reference File을 통해 확인
STR $0, [$2, #0x024];
25번째 bit인 I bit가 0이므로, #으로 표기
➢ Instruction이 어떤 의미를 가지는지 서술
2번 레지스터에 저장된 값에 #0x024값을 더해 주소 값 계산
0번 레지스터에 저장되어 있던 값을 위에서 계산한 메모리 주소에 저장

010 : E3A0003F

➢ instruction을 Binary로 변환
1110 0011 1010 0000 0000 0000 0011 1111
➢ 어떤 Instruction인지 Reference File을 통해 확인
MOV $0, #0x3F;
➢ Instruction이 어떤 의미를 가지는지 서술
0번 레지스터에 0x3F를 저장

011 : E5820028

➢ instruction을 Binary로 변환
1110 0101 1000 0010 0000 0000 0010 1000
➢ 어떤 Instruction인지 Reference File을 통해 확인
STR $0, [$2, #0x028];
25번째 bit인 I bit가 0이므로, #으로 표기
➢ Instruction이 어떤 의미를 가지는지 서술
2번 레지스터에 저장된 값에 #0x028값을 더해 주소 값 계산
0번 레지스터에 저장되어 있던 값을 위에서 계산한 메모리 주소에 저장

012 : E3A00008

➢ instruction을 Binary로 변환
1110 0011 1010 0000 0000 0000 0000 1000
➢ 어떤 Instruction인지 Reference File을 통해 확인
MOV $0, #0x8;

➢ Instruction이 어떤 의미를 가지는지 서술

0번 레지스터에 0x8을 저장

013 : E582002C

➢ instruction을 Binary로 변환

1110 0101 1000 0010 0000 0000 0010 1100

➢ 어떤 Instruction인지 Reference File을 통해 확인

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

25번째 bit인 I bit가 0이므로, #으로 표기

➢ Instruction이 어떤 의미를 가지는지 서술

2번 레지스터에 저장된 값에 #0x02C값을 더해 주소 값 계산

0번 레지스터에 저장되어 있던 값을 위에서 계산한 메모리 주소에 저장

014 : E59F3E9C

➢ instruction을 Binary로 변환

1110 0101 1001 1111 0011 1110 1001 1100

➢ 어떤 Instruction인지 Reference File을 통해 확인

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

25번째 bit인 I bit가 0이므로, #으로 표기

➢ Instruction이 어떤 의미를 가지는지 서술

15번 Register에 저장된 값에 #0xE9C값을 더한 주소의 값을 메모리로부터 읽어 와서 3번 Register에 저장

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

015 : E59F1E9C

➢ instruction을 Binary로 변환

1110 0101 1001 1111 0001 1110 1001 1100

➢ 어떤 Instruction인지 Reference File을 통해 확인

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

25번째 bit인 I bit가 0이므로, #으로 표기

➢ Instruction이 어떤 의미를 가지는지 서술

15번 Register에 저장된 값에 #0xE9C값을 더한 주소의 값을 메모리로부터 읽어 와서 1번 Register에 저장

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

016 : E5831000

➢ instruction을 Binary로 변환

1110 0101 1000 0011 0001 0000 0000 0000

➢ 어떤 Instruction인지 Reference File을 통해 확인

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

➢ Instruction이 어떤 의미를 가지는지 서술

3번 레지스터에 저장된 값에 #0x000값을 더해 주소 값 계산

1번 레지스터에 저장되어 있던 값을 위에서 계산한 메모리 주소에 저장

017 : E59F9E98

➢ instruction을 Binary로 변환
1110 0101 1001 1111 1001 1110 1001 1000
➢ 어떤 Instruction인지 Reference File을 통해 확인
LDR $9, [$15, #0xE98];
25번째 bit인 I bit가 0이므로, #으로 표기
➢ Instruction이 어떤 의미를 가지는지 서술
15번 Register에 저장된 값에 #0xE98값을 더한 주소의 값을 메모리로부터 읽어 와서 9번 Register에 저장
메모리의 [$15 + #0xE98] 주소에 저장된 값을 읽어와 $9에 저장

018 : E3A08000

➢ instruction을 Binary로 변환
1110 0011 1010 0000 1000 0000 0000 0000
➢ 어떤 Instruction인지 Reference File을 통해 확인
MOV $8, #0x0;
➢ Instruction이 어떤 의미를 가지는지 서술
8번 레지스터에 0x0을 저장

019 : E5898000

➢ instruction을 Binary로 변환
1110 0101 1000 1001 1000 0000 0000 0000
➢ 어떤 Instruction인지 Reference File을 통해 확인
STR $8, [$9, #0x000];
➢ Instruction이 어떤 의미를 가지는지 서술
9번 레지스터에 저장된 값에 #0x000값을 더해 주소 값 계산
8번 레지스터에 저장되어있던 값을 위에서 계산한 메모리 주소에 저장

01A : E5898004

➢ instruction을 Binary로 변환
1110 0101 1000 1001 1000 0000 0000 0100
➢ 어떤 Instruction인지 Reference File을 통해 확인
STR $8, [$9, #0x004];
➢ Instruction이 어떤 의미를 가지는지 서술
9번 레지스터에 저장된 값에 #0x004값을 더해 주소 값 계산
8번 레지스터에 저장되어있던 값을 위에서 계산한 메모리 주소에 저장

01B : E5898008

➢ instruction을 Binary로 변환
1110 0101 1000 1001 1000 0000 0000 1000
➢ 어떤 Instruction인지 Reference File을 통해 확인
STR $8, [$9, #0x008];
➢ Instruction이 어떤 의미를 가지는지 서술
9번 레지스터에 저장된 값에 #0x008값을 더해 주소 값 계산
8번 레지스터에 저장되어있던 값을 위에서 계산한 메모리 주소에 저장

01C : E589800C

➢ instruction을 Binary로 변환
1110 0101 1000 1001 1000 0000 0000 1100
➢ 어떤 Instruction인지 Reference File을 통해 확인
STR $8, [$9, #0x00C];
➢ Instruction이 어떤 의미를 가지는지 서술
9번 레지스터에 저장된 값에 #0x00C값을 더해 주소 값 계산
8번 레지스터에 저장되어있던 값을 위에서 계산한 메모리 주소에 저장

01D : E5898010

➢ instruction을 Binary로 변환
1110 0101 1000 1001 1000 0000 0001 0000
➢ 어떤 Instruction인지 Reference File을 통해 확인
STR $8, [$9, #0x010];
➢ Instruction이 어떤 의미를 가지는지 서술
9번 레지스터에 저장된 값에 #0x010값을 더해 주소 값 계산
8번 레지스터에 저장되어있던 값을 위에서 계산한 메모리 주소에 저장

01E : E5898014

➢ instruction을 Binary로 변환
1110 0101 1000 1001 1000 0000 0001 0100
➢ 어떤 Instruction인지 Reference File을 통해 확인
STR $8, [$9, #0x014];
➢ Instruction이 어떤 의미를 가지는지 서술
9번 레지스터에 저장된 값에 #0x014값을 더해 주소 값 계산
8번 레지스터에 저장되어있던 값을 위에서 계산한 메모리 주소에 저장

01F : E5898018

➢ instruction을 Binary로 변환
1110 0101 1000 1001 1000 0000 0001 1000
➢ 어떤 Instruction인지 Reference File을 통해 확인
STR $8, [$9, #0x018];
➢ Instruction이 어떤 의미를 가지는지 서술
9번 레지스터에 저장된 값에 #0x018값을 더해 주소 값 계산
8번 레지스터에 저장되어있던 값을 위에서 계산한 메모리 주소에 저장

020 : E59FDE78

➢ instruction을 Binary로 변환
1110 0101 1001 1111 1101 1110 0111 1000
➢ 어떤 Instruction인지 Reference File을 통해 확인
LDR $13, [$15, #0xE78];
25번째 bit인 I bit가 0이므로, #으로 표기
➢ Instruction이 어떤 의미를 가지는지 서술
15번 Register에 저장된 값에 #0xE78값을 더한 주소의 값을 메모리로부터 읽어 와서 13번 Register에 저장
메모리의 [$15 + #0xEC8] 주소에 저장된 값을 읽어와 $13에 저장

021 : E5931200

➢ instruction을 Binary로 변환
1110 0101 1001 0011 0001 0010 0000 0000

➢ 어떤 Instruction인지 Reference File을 통해 확인

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

25번째 bit인 I bit가 0이므로, #으로 표기

➢ Instruction이 어떤 의미를 가지는지 서술

3번 Register에 저장된 값에 #0x200값을 더한 주소의 값을 메모리로부터 읽어 와서 1번 Register에 저장

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

022 : E3510001

➢ instruction을 Binary로 변환
1110 0011 0101 0001 0000 0000 0000 0001

➢ 어떤 Instruction인지 Reference File을 통해 확인

CMP $1 #0x001

➢ Instruction이 어떤 의미를 가지는지 서술

두 피연산자를 비교하는 작업을 한다. 첫 번째 인자값에서 두 번째 인자 값을 빼는 식으로 비교를 한다. ZF에만 영향을 줌.

023 : 0A000000

➢ instruction을 Binary로 변환
0000 1010 0000 0000 0000 0000 0000 0000

➢ 어떤 Instruction인지 Reference File을 통해 확인

BEQ #0

➢ Instruction이 어떤 의미를 가지는지 서술

branch to label if zero flag is set

37(10진수)번째로 브랜치

024 : EAFFFFFB

➢ instruction을 Binary로 변환
1110 1010 1111 1111 1111 1111 1111 1011

➢ 어떤 Instruction인지 Reference File을 통해 확인

B #-5

➢ Instruction이 어떤 의미를 가지는지 서술

1. Sign-extending the 24-bit signed immediate to 30 bits

2. Shifting the result left two bits to form a 32-bit value

Adding this to the contents of the PC, which contains the address of the branch instruction plus 8 bytes.

현재 명령어의 주소 (36*4)+8에 위에서 계산한 -20을 더해 132를 만듦

132를 4로 나누어 Word 단위의 명령어 순서 중 33번째로 Branch함

동작 순서

000->008->009->00A->00B->00C->00D->00E->00F->010->011->012->013->014->015->
016->017->018->019->01A->01B->01C->01D->01E->01F->020->021->
022(Zero Flag가 1로 설정된 경우)->023->025(끝)
022(Zero Flag가 0인 경우)->023->024->021(021부터 024까지 무한반복)