# 8-3 컴퓨터 설계 기초

## p. 437

### 단순 컴퓨터 아키텍처

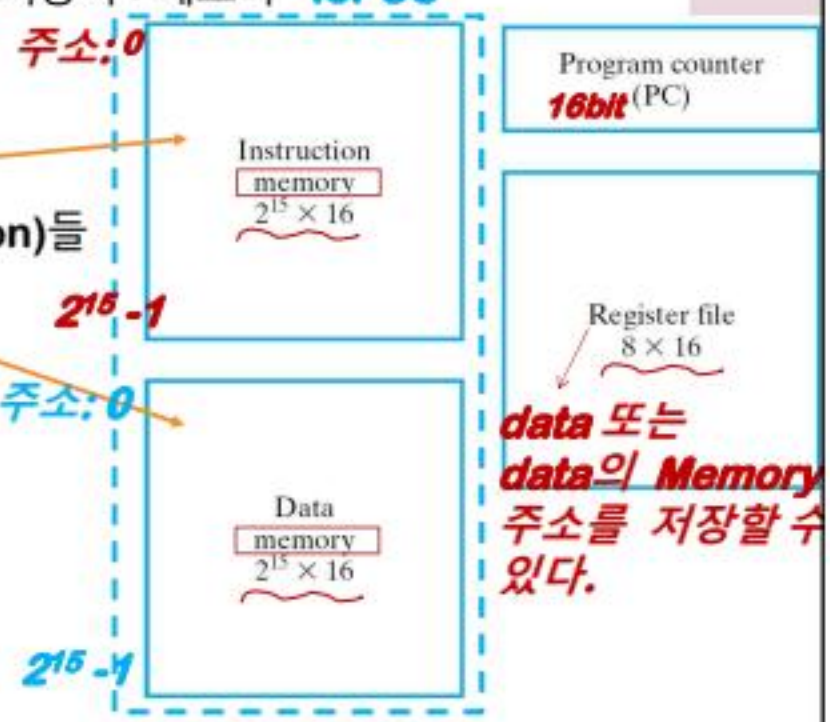
단순 컴퓨터 아키텍처의 제어장치를 이해한다.
프로그램 가능한 시스템에 대한 제어설계
명령어 ➔ 마이크로 연산으로 실현





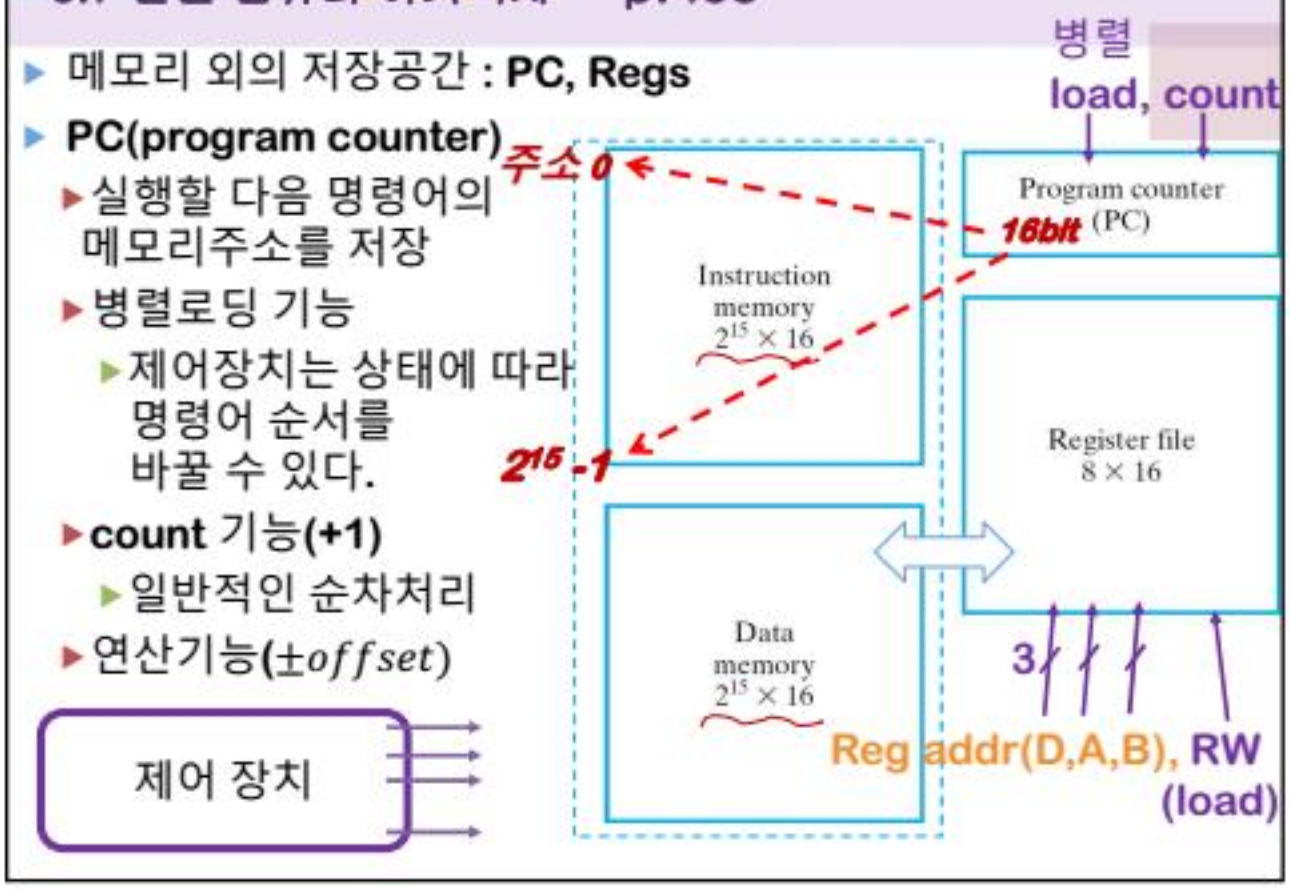
## 8.7 단순 컴퓨터(SC : Simple computer) 아키텍처 p.437

- 이해를 돕기 위한 단순한 구조
  - 데이터 처리 장치+제어장치+메모리
- **Memory**의 구성
  - 프로그램
    - 명령어(**instruction**)들
  - **data** 저장



## 8.7 단순 컴퓨터 아키텍처 p.438

- 메모리 외의 저장공간 : **PC, Regs**
- **PC(program counter)**
  - 실행할 다음 명령어의 메모리주소를 저장
  - 병렬로딩 기능
    - 제어장치는 상태에 따라 명령어 순서를 바꿀 수 있다.
  - **count** 기능**(+1)**
    - 일반적인 순차처리
  - 연산기능$(\pm offset)$



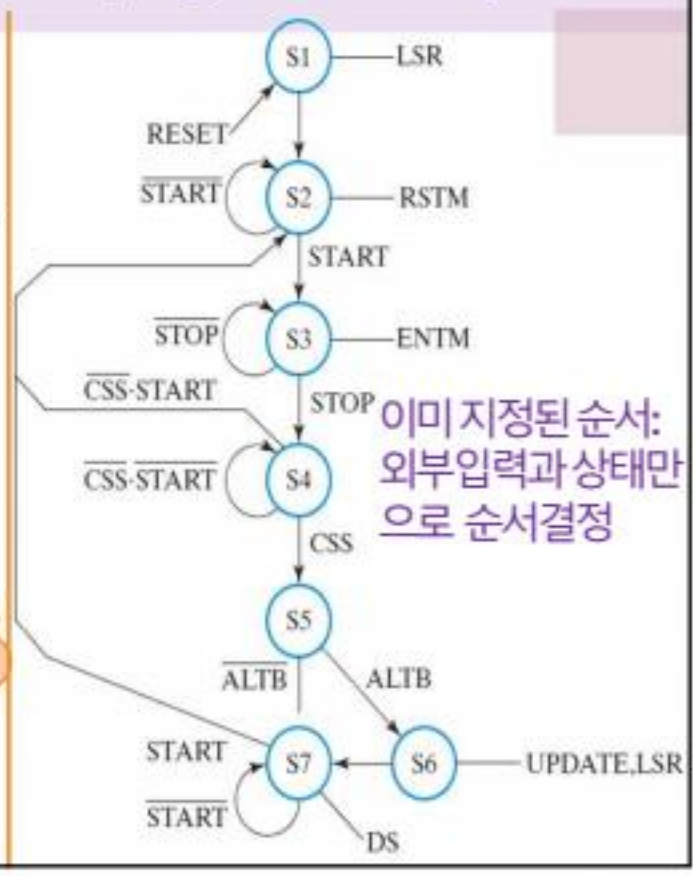

3

## programmable System / nonprogrammable System

- 제어 장치의 비교

- **programmable System**
  - **메모리에 명령어들을 둔다**
  - 제어장치가
    - 명령어 해석
    - 순서결정
    - 제어 워드 발생(실행)

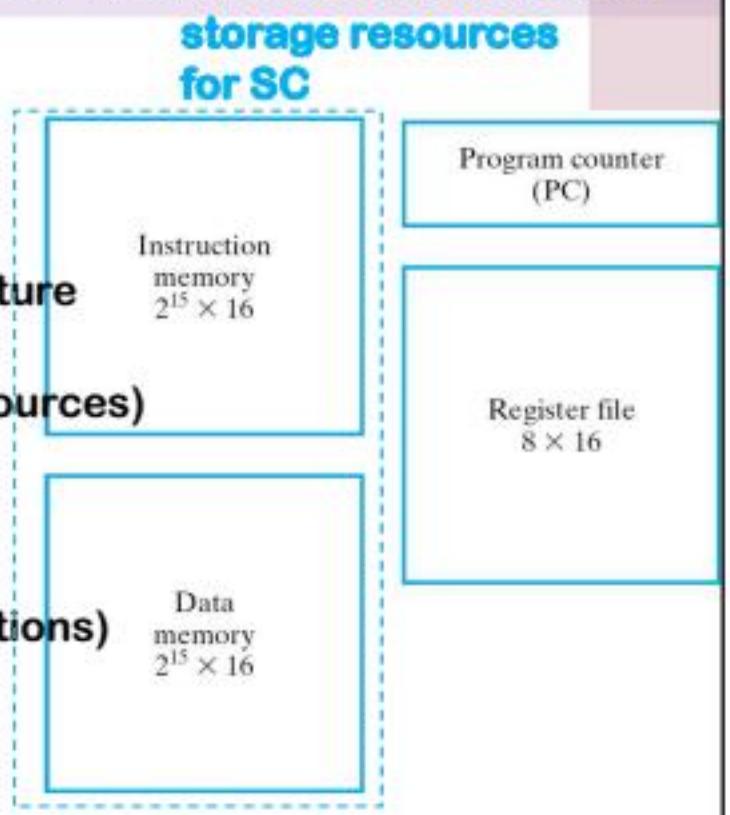
이 과정을 살펴보자.




4

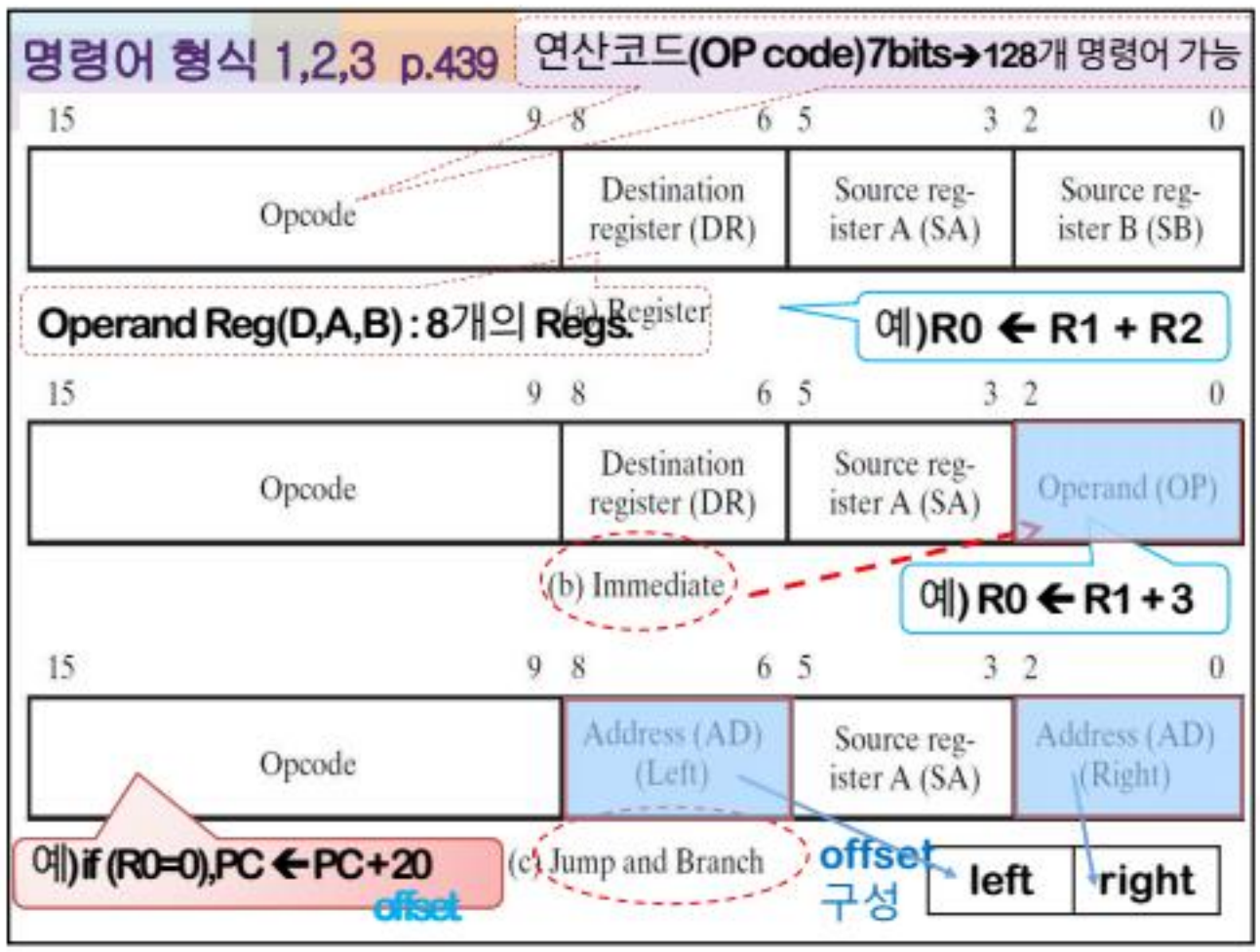
## 명령어 셋 (Instruction Set) Architecture (ISA) p.439

- 명령어(instruction)
- **Instruction Set**
  - 한 컴퓨터의 수행 가능한 명령어 집합
- **Instruction Set Architecture** 의 **3**가지 주요요소
  1. 저장 자원**(storage resources)**
  2. 명령어 형식 **(instruction formats)**
  3. 명령어 명세 **(instruction specifications)**
     각 명령에 대해 상세히 기술한 것




5

## 명령어 형식 1,2,3 p.439




6

3

## 명령어 형식(a)　명령어 명세　p.443

▶ **operation code에 의해 형식(a)(b)(c) 이 결정된다.**

실행 후 유효한 상태비트

(a)형식

SB불필요

| Instruction | Opcode | Mnemonic | Format | Description | Status Bits |
|---|---|---|---|---|---|
| Move A | 0000000 | MOVA | RD, RA | R[DR] ← R[SA]* | N, Z |
| Increment | 0000001 | INC | RD, RA | R[DR] ← R[SA] + 1* | N, Z |
| Add | 0000010 | ADD | RD, RA, RB | R[DR] ← R[SA] + R[SB]* | N, Z |
| Subtract | 0000101 | SUB | RD, RA, RB | R[DR] ← R[SA] − R[SB]* | N, Z |
| Decrement | 0000110 | DEC | RD, RA | R[DR] ← R[SA] − 1* | N, Z |
| AND | 0001000 | AND | RD, RA, RB | R[DR] ← R[SA] ∧ R[SB]* | N, Z |
| OR | 0001001 | OR | RD, RA, RB | R[DR] ← R[SA] ∨ R[SB]* | N, Z |
| Exclusive OR | 0001010 | XOR | RD, RA, RB | R[DR] ← R[SA] ⊕ R[SB]* | N, Z |
| NOT | 0001011 | NOT | RD, RA | R[DR] ← $\overline{R[SA]}$ * | N, Z |
| Move B | 0001100 | MOVB | RD, RB | R[DR] ← R[SB]* | |
| Shift Right | 0001101 | SHR | RD, RB | R[DR] ← sr R[SB]* | |
| Shift Left | 0001110 | SHL | RD, RB | R[DR] ← sl R[SB]* | |

SA불필요

*:PC←PC+1

7

7

## 명령어 형식(a)　명령어 명세　p.443

▶ **operation code에 의해 형식(a)(b)(c) 이 결정된다.**

(a)형식

| Instruction | Opcode | Mnemonic | Format | Description | Status Bits |
|---|---|---|---|---|---|
| Move A | 0000000 | MOVA | RD, RA | R[DR] ← R[SA]* | N, Z |
| Increment | 0000001 | INC | RD, RA | R[DR] ← R[SA] + 1* | N, Z |
| Add | 0000010 | ADD | RD, RA, RB | R[DR] ← R[SA] + R[SB]* | N, Z |
| Subtract | 0000101 | SUB | RD, RA, RB | R[DR] ← R[SA] − R[SB]* | N, Z |
| Decrement | 0000110 | DEC | RD, RA | R[DR] ← R[SA] − 1* | N, Z |
| AND | 0001000 | AND | RD, RA, RB | R[DR] ← R[SA] ∧ R[SB]* | N, Z |
| OR | 0001001 | OR | RD, RA, RB | R[DR] ← R[SA] ∨ R[SB]* | N, Z |
| Exclusive OR | 0001010 | XOR | RD, RA, RB | R[DR] ← R[SA] ⊕ R[SB]* | N, Z |
| NOT | 0001011 | NOT | RD, RA | R[DR] ← $\overline{R[SA]}$ * | N, Z |
| Move B | 0001100 | MOVB | RD, RB | R[DR] ← R[SB]* | |
| Shift Right | 0001101 | SHR | RD, RB | R[DR] ← sr R[SB]* | |
| Shift Left | 0001110 | SHL | RD, RB | R[DR] ← sl R[SB]* | |

0000101 001 010 011　R1← R2-R3

SUB R1, R2, R3

0000001 100 001 000 XXX　R4← R1+1

INC R4, R1

8

8

4

## 명령어 형식(a)  명령어 명세  p.443

(a)형식

| Instruction | Opcode | Mne-monic | Format | Description | Status Bits |
|---|---|---|---|---|---|
| Load | 0010000 | LD | RD, RA | R[DR] ← M[SA]* | |
| Store | 0100000 | ST | RA, RB | M[SA] ← R[SB]* | |

가정) reg 에 메모리 주소를 저장

0010000 110 010 000 — R6← M[R2]

LD R6, R2 (XXX)





## 명령어 형식(b)  명령어 명세  p.443

(b)형식

| Instruction | Opcode | Mne-monic | Format | Description | Status Bits |
|---|---|---|---|---|---|
| Load Immediate | 1001100 | LDI | RD, OP | R[DR] ← zf OP* | |
| Add Immediate | 1000010 | ADI | RD, RA, OP | R[DR] ← R[SA] + zf OP* | N, Z |

상수 — zero fill

XXX

1001100 001 000 011 — R1← 3

LDI R1, 3

1000010 100 001 110 — R4← R1+6

ADI R4, R1, 6

즉시데이터에 zero fill을 적용한 후 Reg에 저장.

## 명령어 형식(c) 명령어 명세 p.443

(c)형식

| Instruction | Opcode | Mne-monic | Format | Description | Status Bits |
|---|---|---|---|---|---|
| Branch on Zero | 1100000 | BRZ | RA, AD | if (R[SA] = 0) PC ← PC + se AD, N, Z<br>if (R[SA] ≠ 0) PC ← PC + 1 | |
| Branch on Negative | 1100001 | BRN | RA, AD | if (R[SA] < 0) PC ← PC + se AD, N, Z<br>if (R[SA] ≥ 0) PC ← PC + 1 | |
| Jump | 1110000 | JMP | RA | PC ← R[SA] | |

offset

sign extend 0000 0000 0001 1100

1100000 011 001 100
BRZ R1, 28

if(R1=0) PC ← PC + 28(011 100)
otherwise PC ← PC+1

sign extend 1111 1111 1111 1101

1100001 111 010 101
BRN R2, -3

if(R2<0) PC ← PC - 3(111101)
otherwise PC ← PC+1

1110000 000 011 000
JMP R3 XXX XXX

PC ← R3 가정)R3에 주소가 있다.

11

11

## 메모리 내에 저장된 명령어들 p.443

| Decimal Address | Memory Contents | Decimal Opcode | Other Fields | Operation |
|---|---|---|---|---|
| 25 | 0000101 001 010 011 | 5 (Subtract) | DR:1, SA:2, SB:3 | R1 ← R2 – R3 |
| 35 | 0100000 000 100 101 | 32 (Store) | SA:4, SB:5 | M[R4] ← R5 |
| 45 | 1000010 010 111 011 | 66 (Add Immediate) | DR:2, SA:7, OP:3 | R2 ← R7 + 3 |
| 55 | 1100000 101 110 100 | 96 (Branch on Zero) | AD: 44, SA:6 | If R6 = 0, PC ← PC – 20 |
| 70 | 0000000011000000 | Data = 192. After execution of instruction in 35, Data = 80. | | |

PC값 1씩 증가

가정) R4 70 R5 80

상수 3

가정) R6 0

BRZ명령:R6를 ALU에 통과시켜 0인지 판단

이후 PC:35로 바뀔 수 있다.(branch)

101100(-20)

Z상태값으로 판단

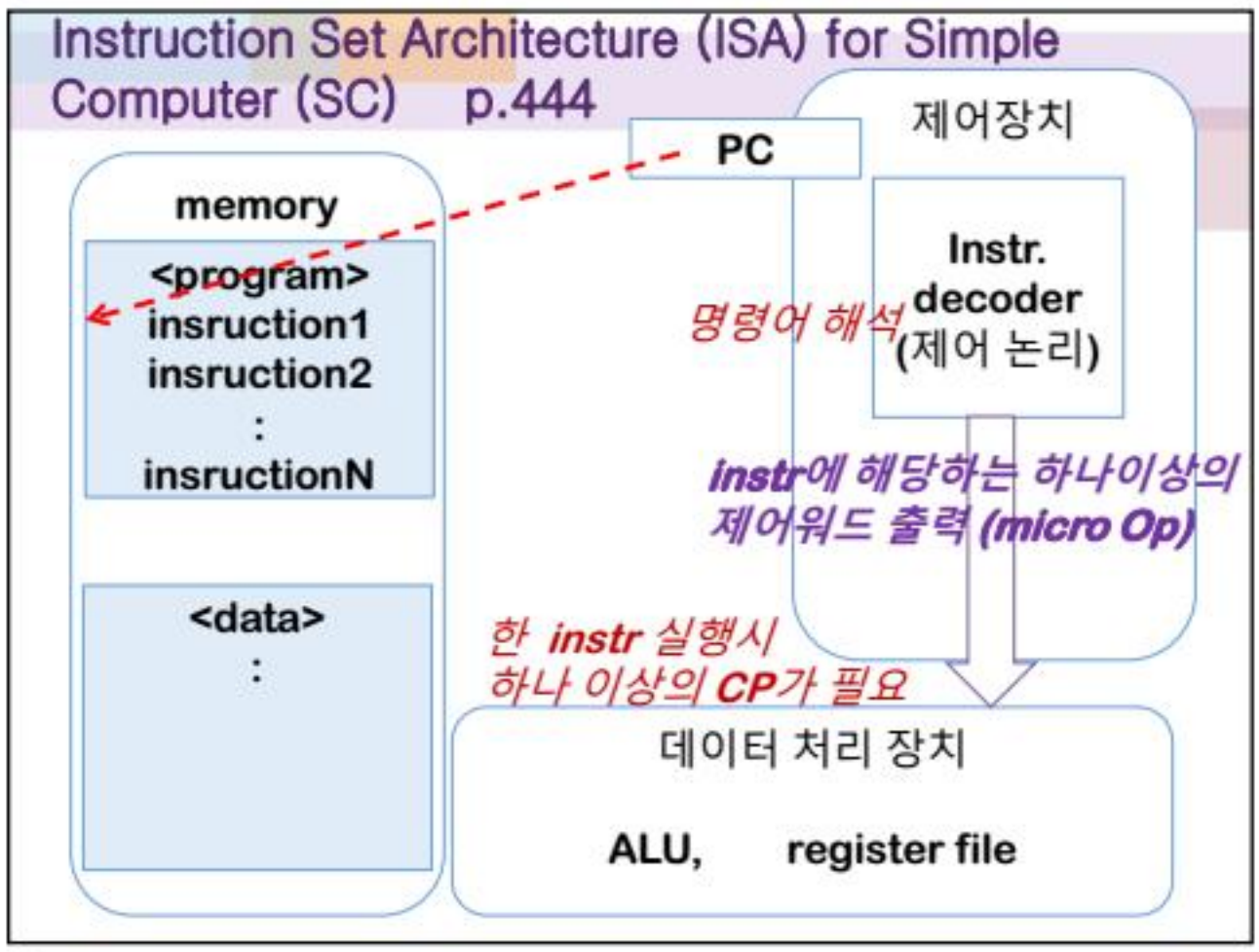
실행 전 저장값

실행 후? 0000 0000 0101 0000
메모리70번지←80저장.

12

6

## Instruction Set Architecture (ISA) for Simple Computer (SC) p.444




13

## SC 의 확장 적용 p.444

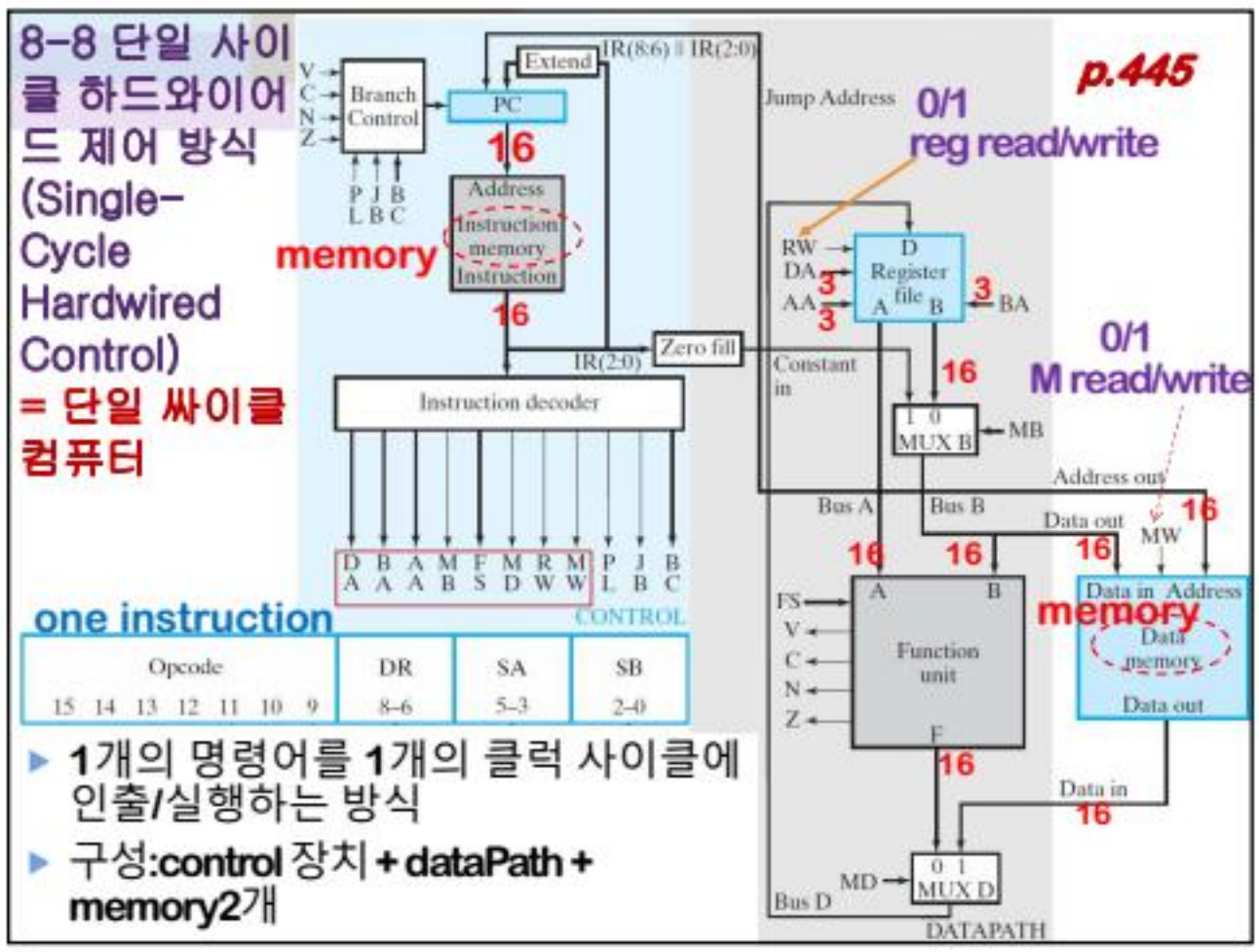
- 메모리 확장 ➔ **PC, Regs** 의 확장 ( **32~64bit** 로 )
- **instruction set**의 확장 ➔ 명령어 형식의 **op code bits**크기 ↑
- 레지스터 개수의 증가 ➔ 명령어 형식의 **DR, SA,SB bits** ↑

지금부터
**SC**의 단순 제어장치를 이해하기 위해
모든 명령어가 **1clock**에 인출/실행하는
**단일 사이클 제어방식**에 대해 공부한다.

(가정)
$2^{16}$ **words**의 작은 고속 메모리

14

14



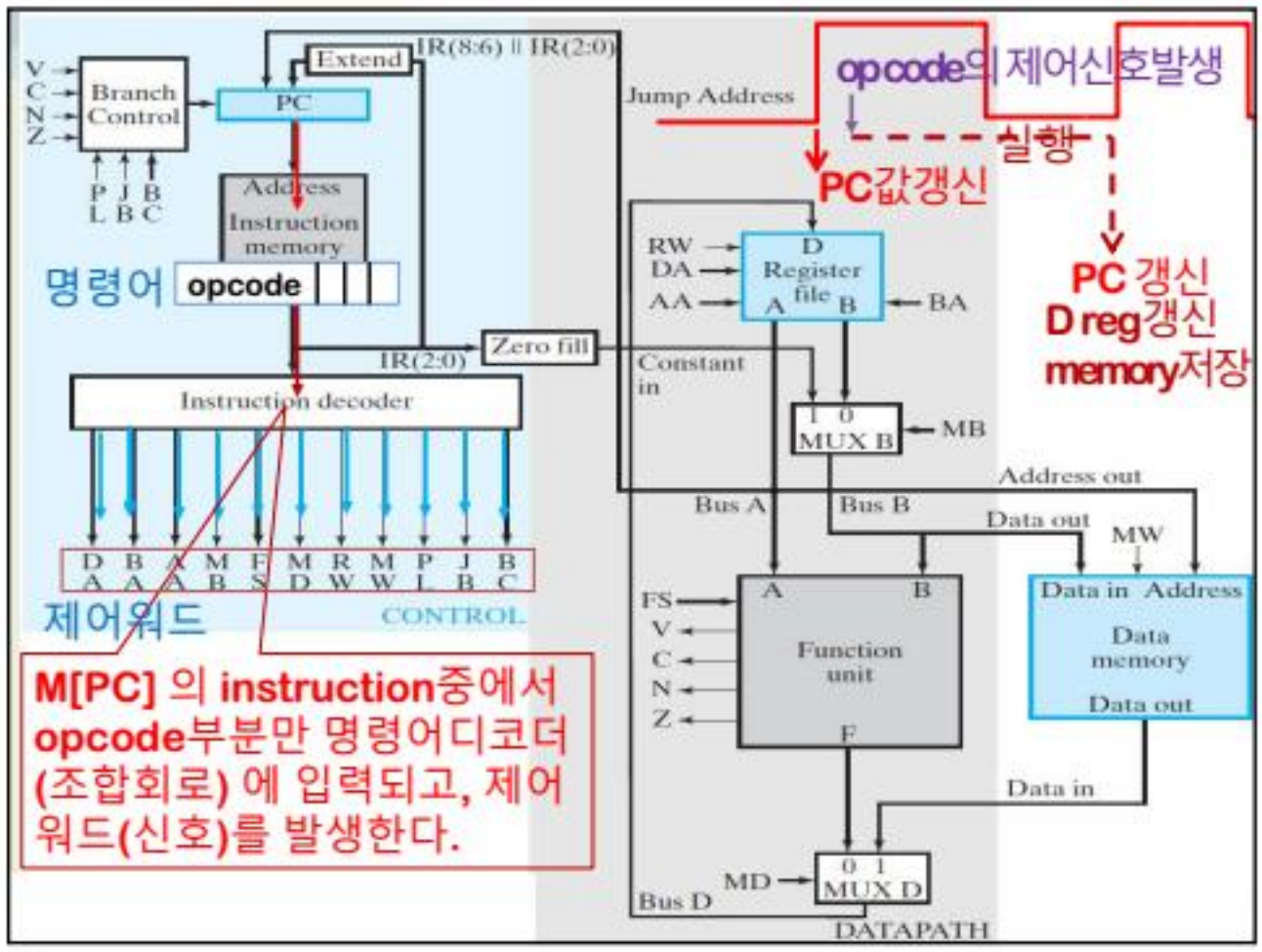

15

## 제안하는 단일 싸이클 컴퓨터의 특징

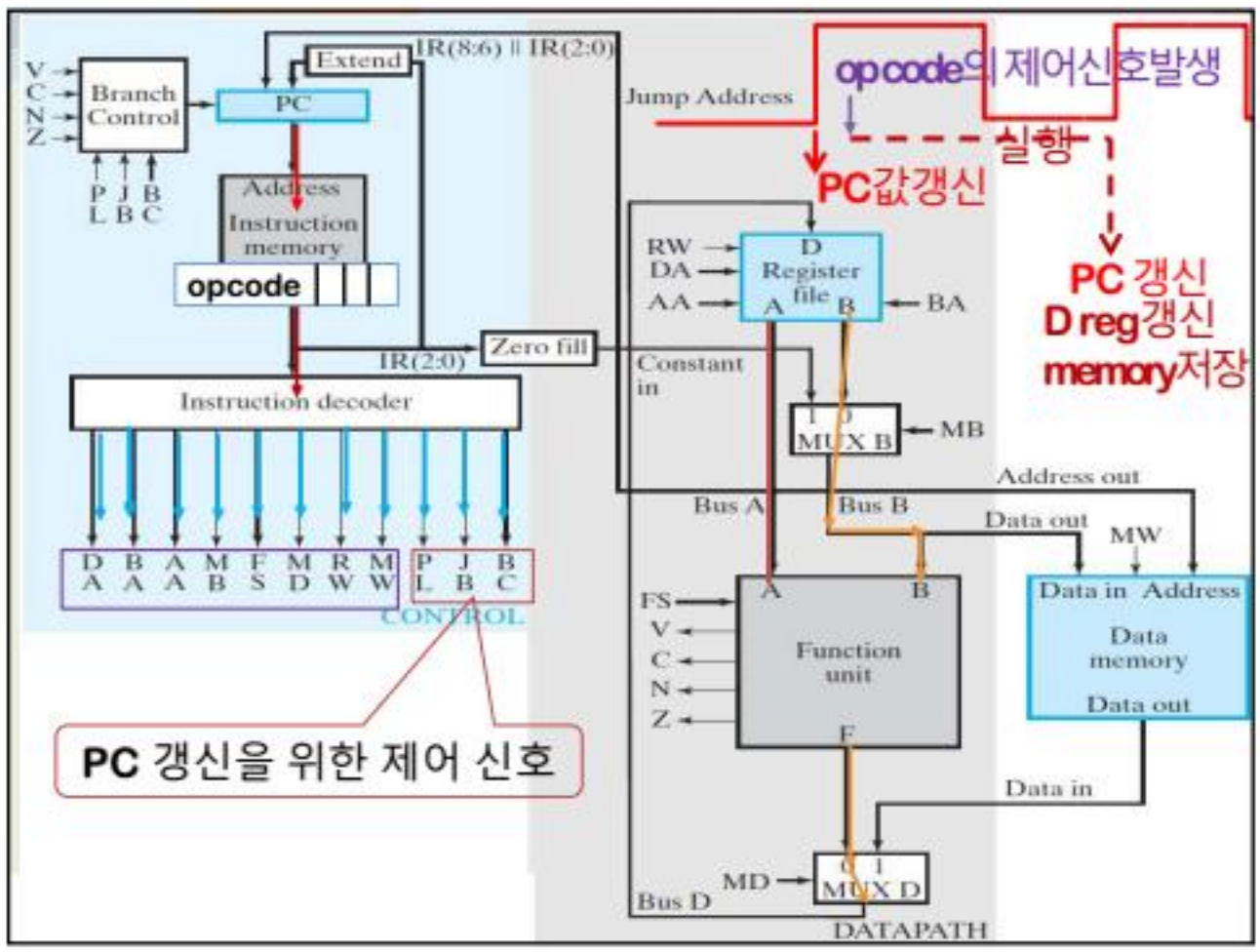
- 명령어 메모리, 데이터 메모리를 **구분하여 단순화했다.**
- 명령어 메모리의 주소는 **PC**가 가리킨다.
  - **pc**에 의해 다음 실행할 명령어를 읽어, 명령어 **decoder**로 전달한다.
- 데이터 메모리의 주소는 **Reg**가 가리킨다.
  - 예) **LD R0 ← M[R3]**
  - **ST M[R2] ← R0**


16

8

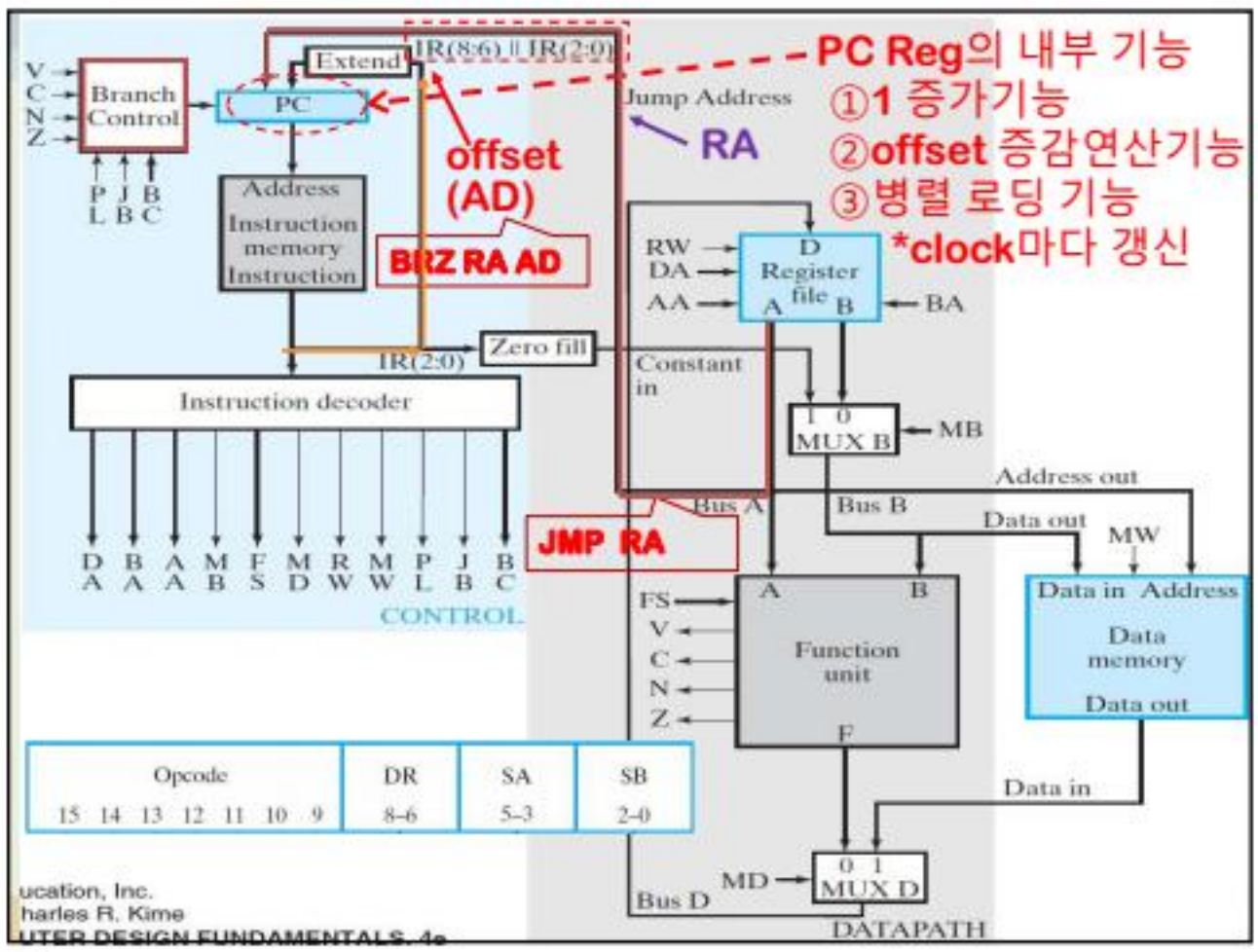

V
C
N
Z
Branch Control
P L J B B C
PC
Extend
IR(8:6) || IR(2:0)
Address
Instruction memory
명령어
opcode
IR(2:0)
Zero fill
Instruction decoder
D A B A A A M B F S M D R W M W P L J B B C
제어워드
CONTROL
M[PC] 의 instruction중에서 opcode부분만 명령어디코더 (조합회로) 에 입력되고, 제어 워드(신호)를 발생한다.
Jump Address
opcode의 제어신호발생
실행
PC값갱신
RW
DA
AA
D
Register file
A
B
BA
Constant in
1 0
MUX B
MB
Address out
Bus A
Bus B
Data out
MW
FS
V
C
N
Z
A
B
Function unit
F
Data in Address
Data memory
Data out
PC 갱신
D reg갱신
memory저장
Data in
MD
0 1
MUX D
Bus D
DATAPATH

17

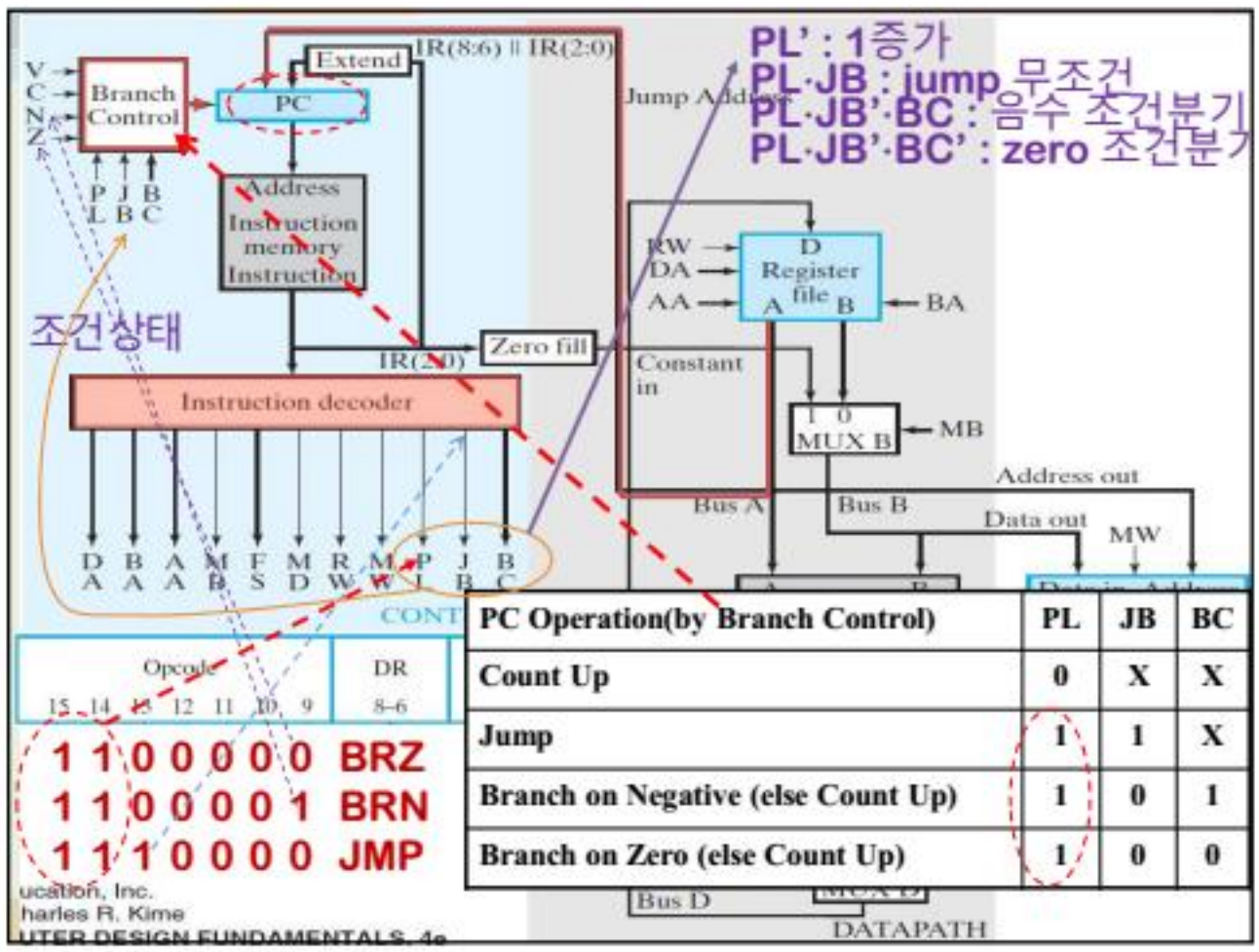

V
C
N
Z
Branch Control
P L J B B C
PC
Extend
IR(8:6) || IR(2:0)
Address
Instruction memory
opcode
IR(2:0)
Zero fill
Instruction decoder
D A B A A A M B F S M D R W M W P L J B B C
CONTROL
PC 갱신을 위한 제어 신호
Jump Address
opcode의 제어신호발생
실행
PC값갱신
RW
DA
AA
D
Register file
A
B
BA
Constant in
MB
Address out
Bus A
Bus B
Data out
MW
FS
V
C
N
Z
A
B
Function unit
F
Data in Address
Data memory
Data out
PC 갱신
D reg갱신
memory저장
Data in
MD
Bus D
DATAPATH

18

9



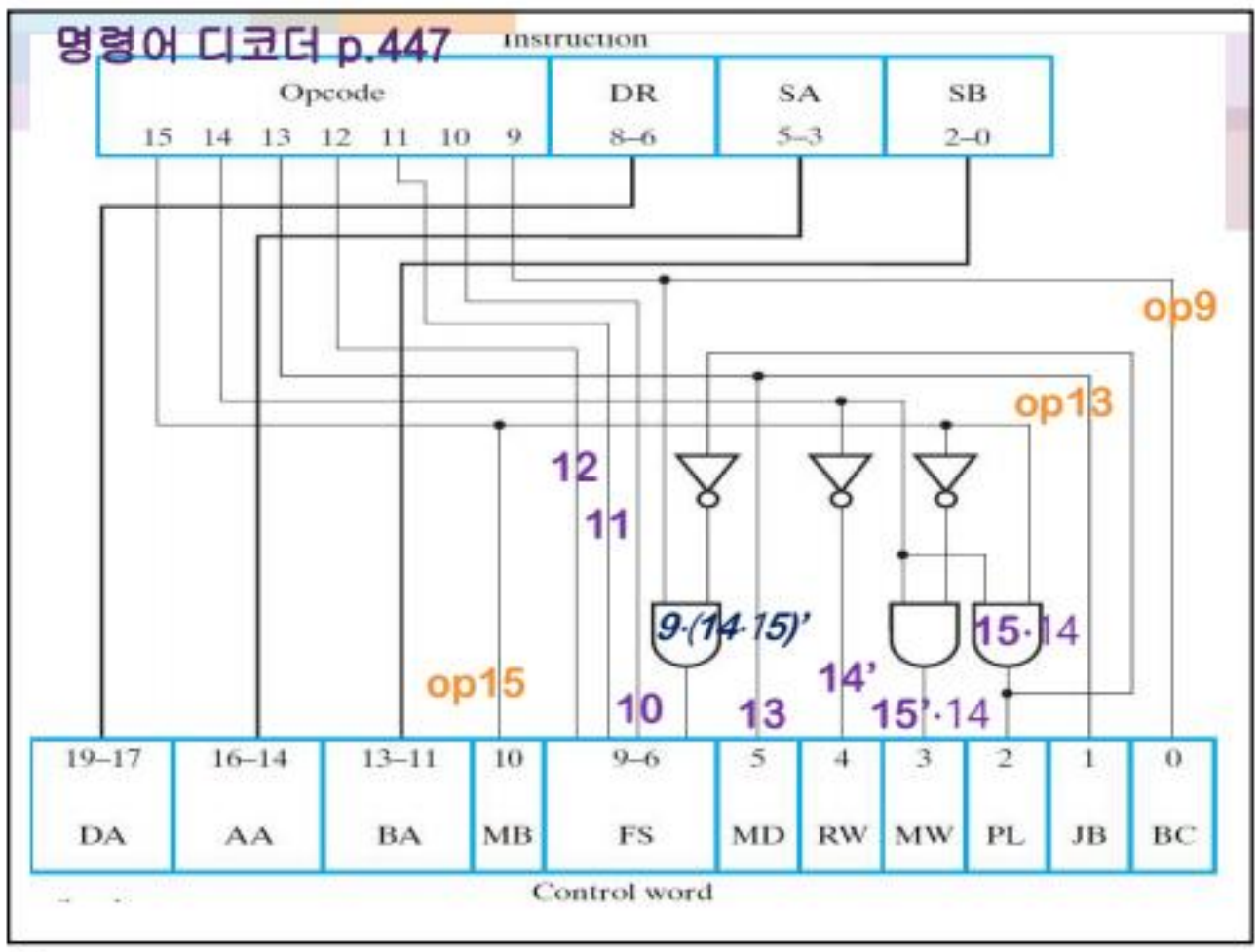

19




| PC Operation(by Branch Control) | PL | JB | BC |
|---|---|---|---|
| Count Up | 0 | X | X |
| Jump | 1 | 1 | X |
| Branch on Negative (else Count Up) | 1 | 0 | 1 |
| Branch on Zero (else Count Up) | 1 | 0 | 0 |

20

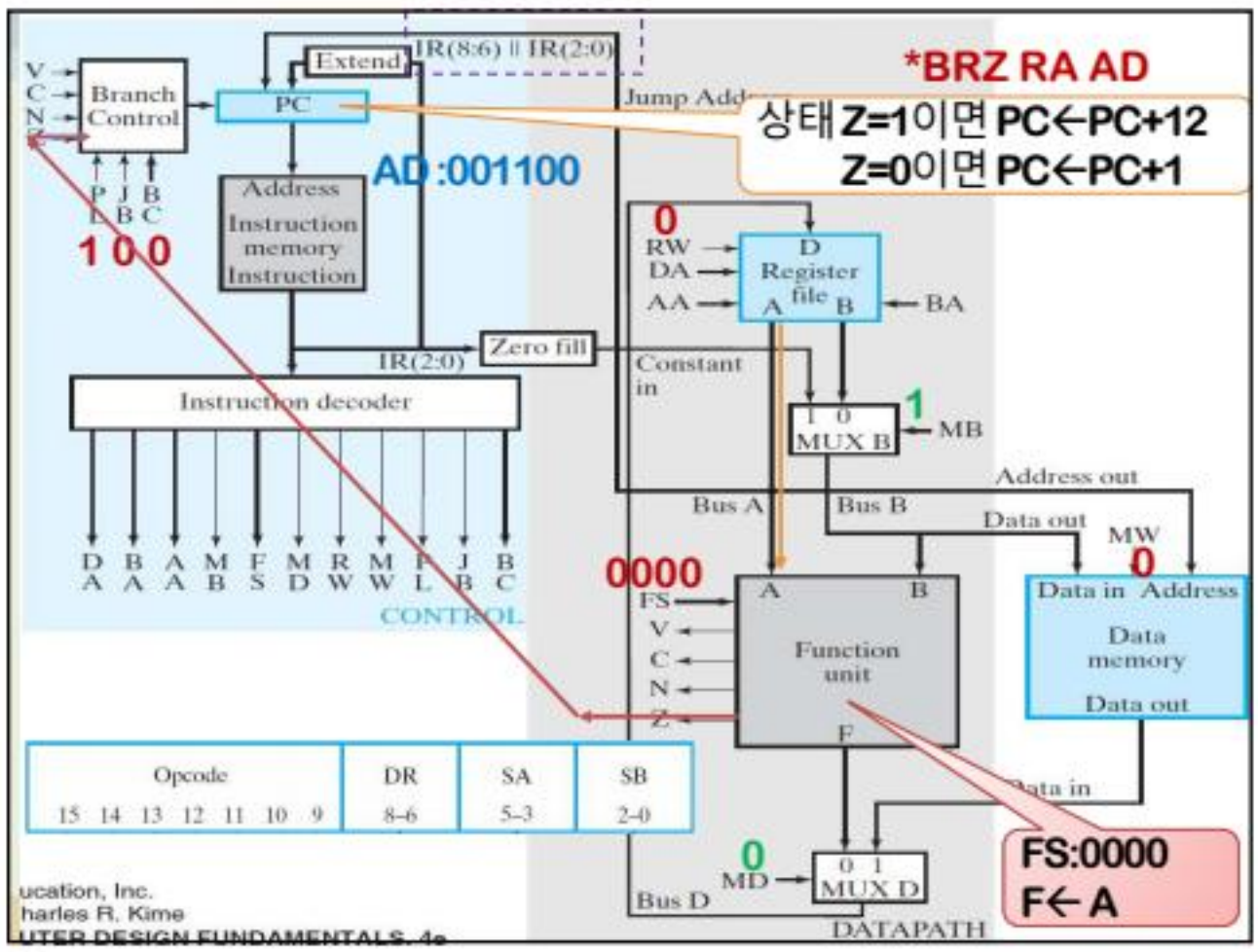
10


명령어 디코더 p.447
Instruction
Opcode
15 14 13 12 11 10 9
DR
8–6
SA
5–3
SB
2–0
op9
op13
12
11
9·(14·15)'
15·14
op15
10
13
14'
15'·14
19–17
DA
16–14
AA
13–11
BA
10
MB
9–6
FS
5
MD
4
RW
3
MW
2
PL
1
JB
0
BC
Control word

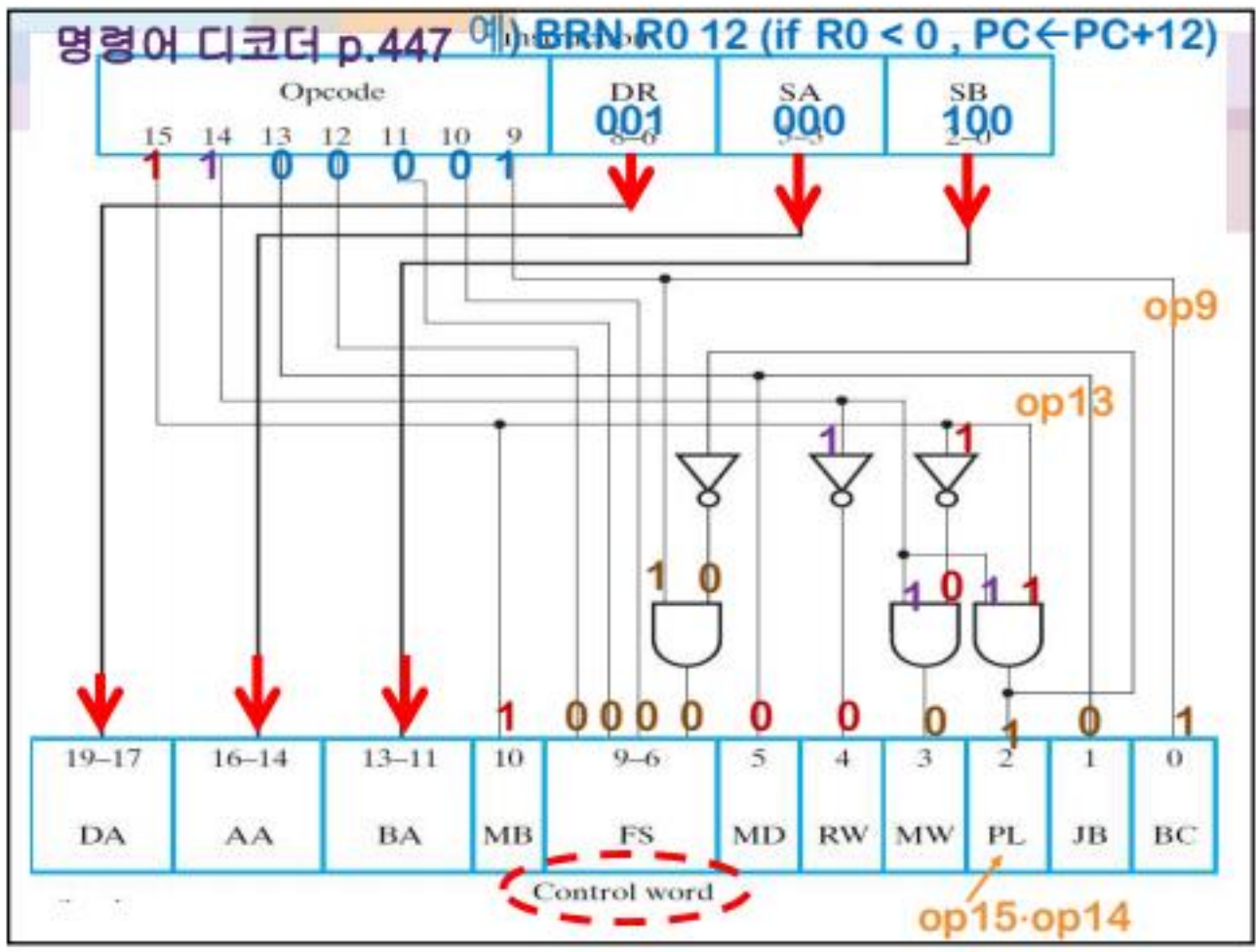
21


명령어 디코더 p.447
예) BRZ R0 12 (if R0=0, PC←PC+12)
Opcode
15 14 13 12 11 10 9
1 1 0 0 0 0 0
DR
001
SA
000
SB
100
op9
op13
1
1
13
14'
1 0 1 1
9·(14·15)'
15·14
15'·14
1 0 0 0 0 0 0 0 1 0 0
19–17
001
DA
16–14
000
AA
13–11
100
BA
10
MB
9–6
FS
5
MD
4
RW
3
MW
2
PL
1
JB
0
BC
Control word
op15·op14

22

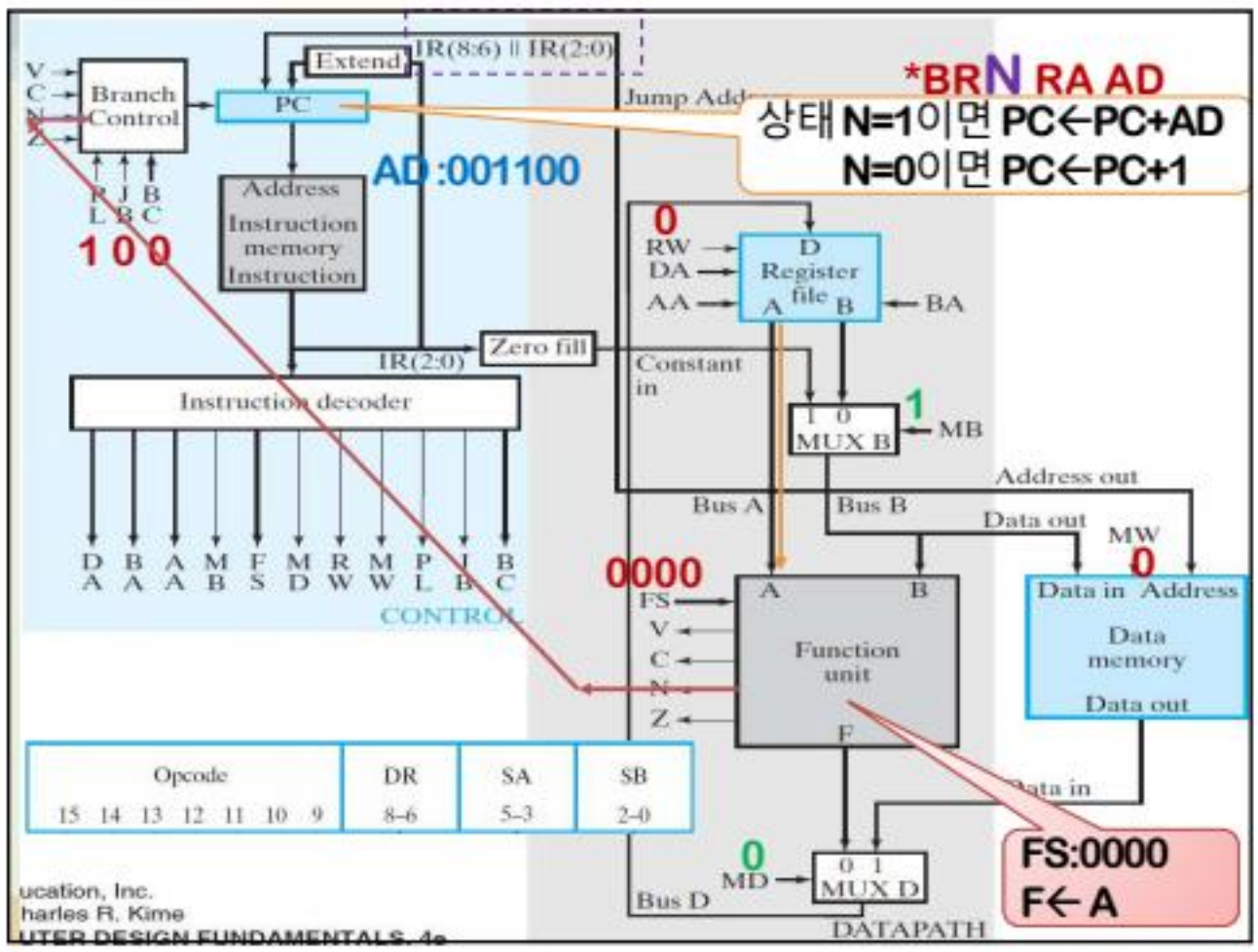

*BRZ RA AD
상태 Z=1이면 PC←PC+12
Z=0이면 PC←PC+1
AD :001100
Extend
IR(8:6) || IR(2:0)
Branch Control
PC
Address
Instruction memory
Instruction
1 0 0
Zero fill
IR(2:0)
Instruction decoder
Constant in
RW
DA
AA
Register file
BA
MB
MUX B
Address out
Bus A
Bus B
Data out
MW
0000
FS
V
C
N
Z
Function unit
Data in Address
Data memory
Data out
Opcode
15 14 13 12 11 10 9
DR
8–6
SA
5–3
SB
2–0
MD
MUX D
Bus D
DATAPATH
FS:0000
F← A
CONTROL

23

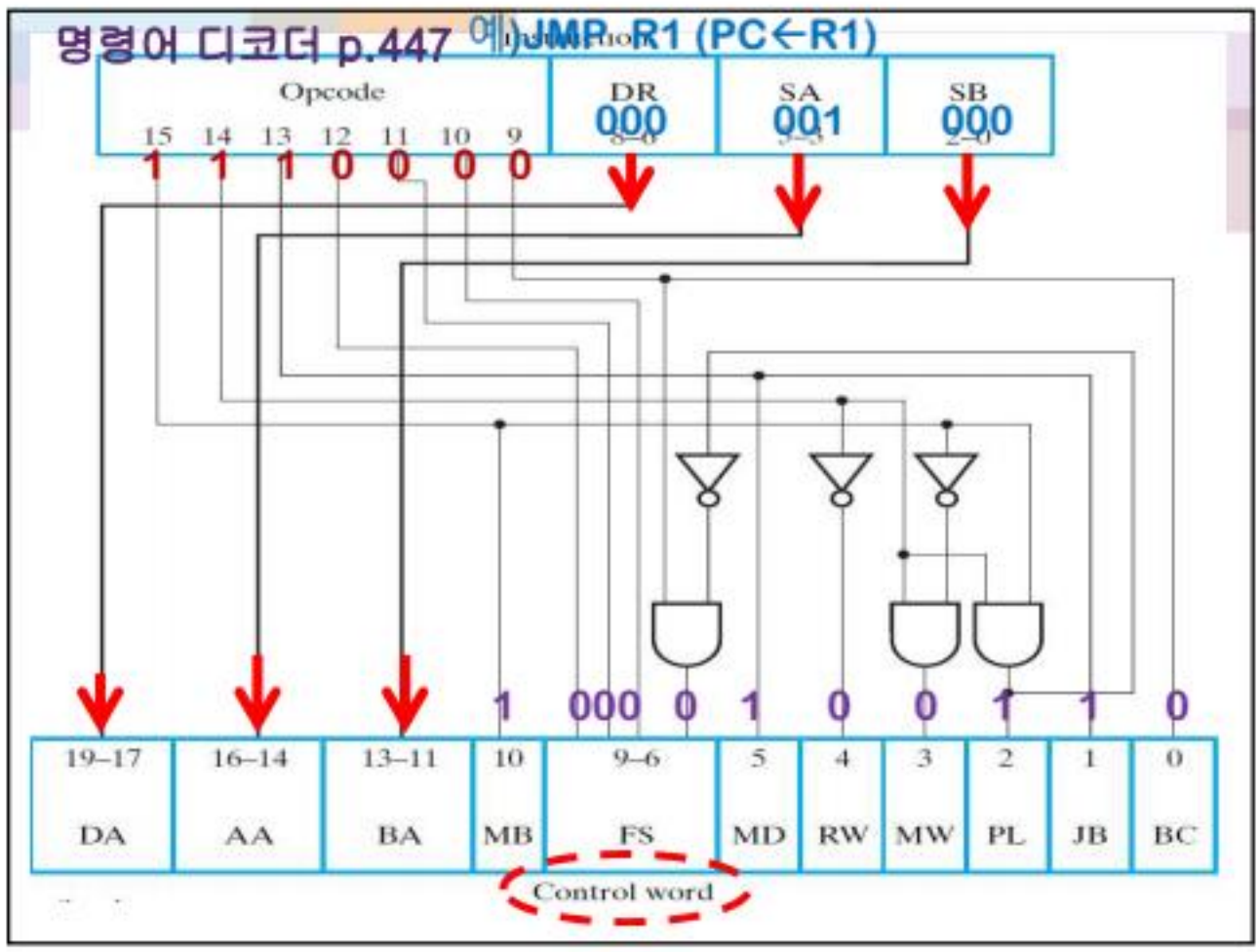

명령어 디코더 p.447
예) BRN R0 12 (if R0 < 0 , PC←PC+12)
Opcode
15 14 13 12 11 10 9
1 1 0 0 0 0 1
DR
001
SA
000
SB
100
op9
op13
19–17
DA
16–14
AA
13–11
BA
10
MB
9–6
FS
5
MD
4
RW
3
MW
2
PL
1
JB
0
BC
1 0 0 0 0 0 0 0 1 0 1
Control word
op15·op14

24

12

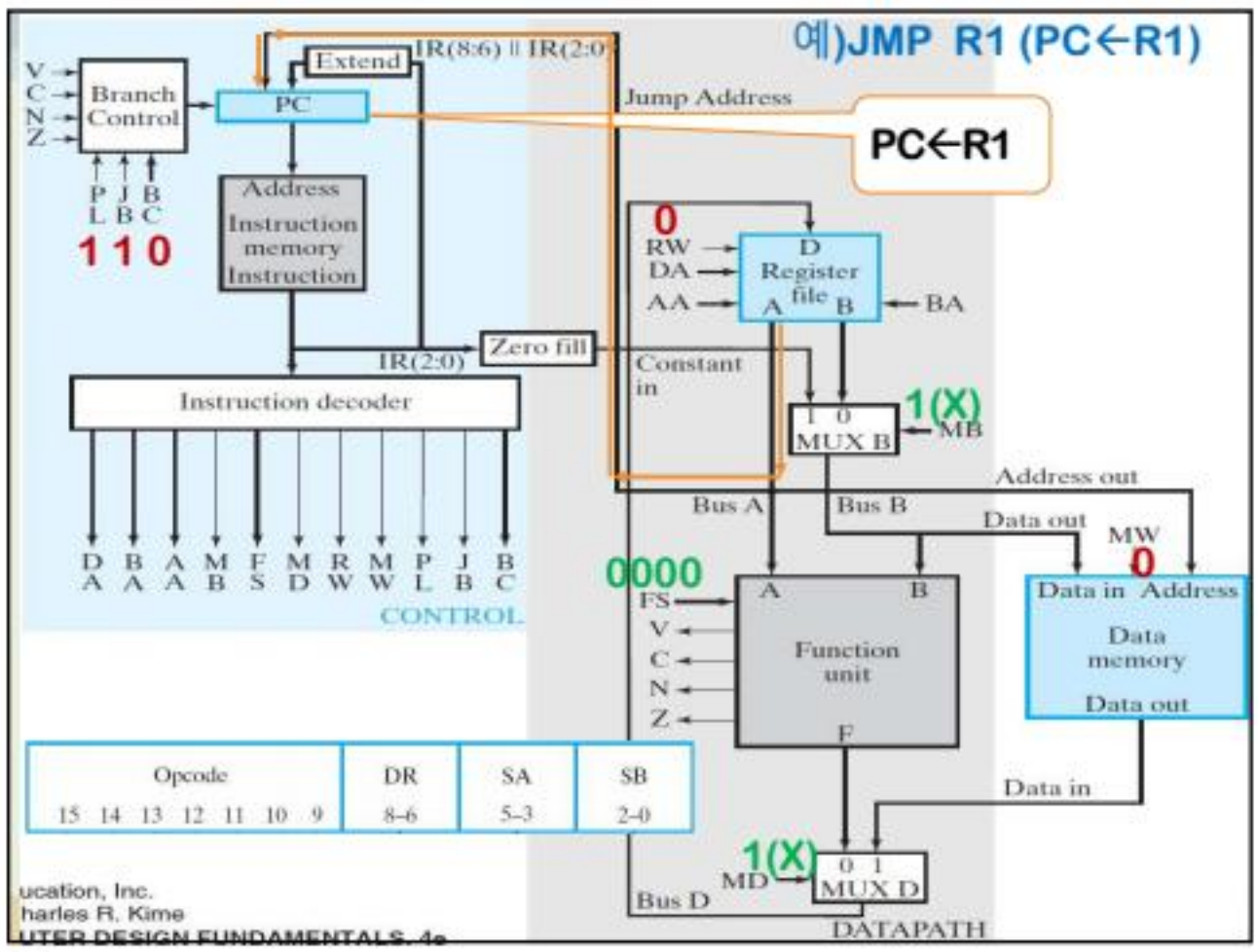

*BRN RA AD
상태 N=1이면 PC←PC+AD
N=0이면 PC←PC+1
AD :001100
V
C
N
Z
Branch Control
P L
J B
B C
1 0 0
PC
Extend
IR(8:6) || IR(2:0)
Jump Add
Address
Instruction memory
Instruction
IR(2:0)
Zero fill
Instruction decoder
D A
B A
A A
M B
F S
M D
R W
M W
P L
J B
B C
CONTROL
0
RW
DA
AA
D
Register file
A
B
BA
Constant in
1
MB
MUX B
Address out
Bus A
Bus B
Data out
MW
0
0000
FS
V
C
N
Z
A
B
Function unit
F
Data in Address
Data memory
Data out
Opcode
15 14 13 12 11 10 9
DR
8–6
SA
5–3
SB
2–0
Data in
0
MD
MUX D
Bus D
DATAPATH
FS:0000
F← A
ucation, Inc.
harles R. Kime
UTER DESIGN FUNDAMENTALS, 4e

25


명령어 디코더 p.447
예) JMP R1 (PC←R1)
Opcode
15 14 13 12 11 10 9
1 1 1 0 0 0 0
DR
000
SA
001
SB
000
1 000 0 1 0 0 1 1 0
19–17
DA
16–14
AA
13–11
BA
10
MB
9–6
FS
5
MD
4
RW
3
MW
2
PL
1
JB
0
BC
Control word

26

13




27

명령어 함수유형1,2,3,4
1.레지스터 간 연산
2.메모리 읽기, 쓰기

명령어 유형을 3개의 비트(15-13)로 결정 (단순 컴퓨터의 설계)

3bit로 구분된다.

| Instruction Function Type | 15 | 14 | 13 | 9 |
|---|---|---|---|---|
| Function-unit operations using registers | 0 | 0 | 0 | X |
| Memory read | 0 | 0 | 1 | X |
| Memory write | 0 | 1 | 0 | X |
| Function-unit operations using register and constant | 1 | 0 | 0 | X |
| Unconditional jump | 1 | 1 | 1 | X |

| Control Word Bits | | | |
|---|---|---|---|
| Move A | 000000 | MOVA | RD, RA |
| Increment | 000001 | INC | RD, RA |
| Add | 000010 | ADD | RD, RA, RB |
| Subtract | 000101 | SUB | RD, RA, RB |
| Decrement | 000110 | DEC | RD, RA |
| AND | 001000 | AND | RD, RA, RB |
| OR | 001001 | OR | RD, RA, RB |
| Exclusive OR | 001010 | XOR | RD, RA, RB |
| NOT | 001011 | NOT | RD, RA |
| Move B | 001100 | MOVB | RD, RB |
| Shift Right | 001101 | SHR | RD, RB |
| Shift Left | 001110 | SHL | RD, RB |
| Load | 0010000 | LD | RD, RA |
| Store | 0100000 | ST | RA, RB |

28

## 명령어 함수 유형3,4

3.reg와 상수간 연산

4.조건분기, 점프

| Instruction Function Type | 15 | 14 | 13 | 9 | MB | MD | RW | MW | PL | JB | BC |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Function-unit operations using registers | 0 | Load Immediate | | | 1 | 0 | 0 | 1 | 0 | 0 | RD, OP |
| | 0 | Add Immediate | | | 1 | 0 | 0 | 0 | 1 | 0 | RD, RA, OP |
| Memory read | 0 | 0 | 1 | X | 0 | 1 | 1 | 0 | 0 | X | X |
| Memory write | 0 | 1 | 0 | X | 0 | X | 0 | 1 | 0 | X | X |
| Function-unit operations using register and constant | 1 | 0 | 0 | X | Branch on Zero | | | 1 | 1 | 0 | BRZ |
| Conditional branch on zero ($Z$) | 1 | 1 | 0 | 0 | | | | | | | |
| Conditional branch on negative ($N$) | 1 | 1 | 0 | 1 | Branch on Negative | | | 1 | 1 | 0 | BRN |
| Unconditional jump | 1 | 1 | 1 | X | Jump | | | 1 | 1 | 1 | JMP |

29

## 명령어 디코더 p.447

예)movA r0 r2 (r0 ← r2)

Instruction

Opcode: 0 0 0 0 0 0 0 (15 14 13 12 11 10 9) | DR: 000 | SA: 010 | SB: 000

op12

op11

op10

op9·PL'

0 000 0 0 1 0 0 0 0

| 19–17 | 16–14 | 13–11 | 10 | 9–6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| DA | AA | BA | MB | FS | MD | RW | MW | PL | JB | BC |

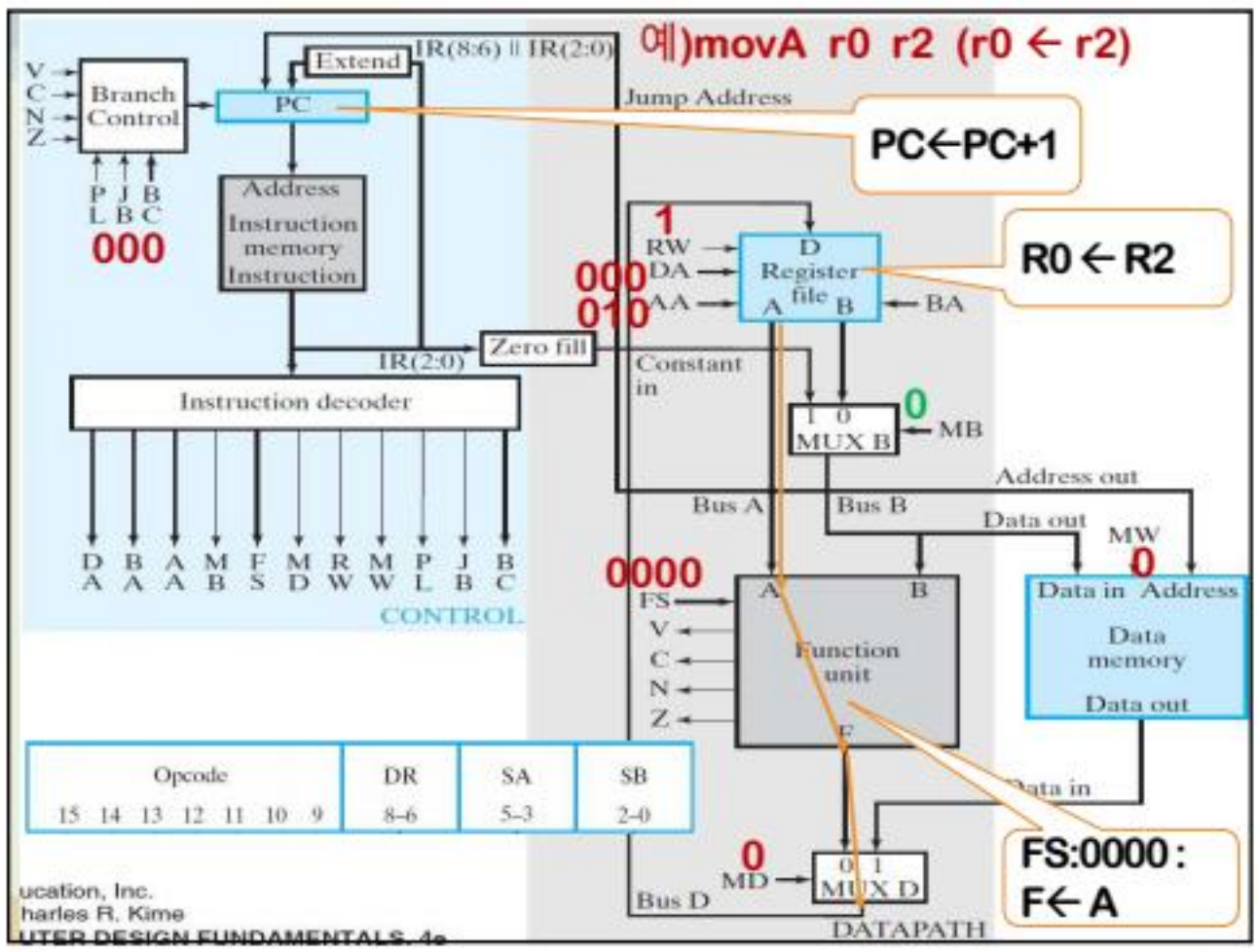
Control word

op9만 FS6에 연결하면 될까?

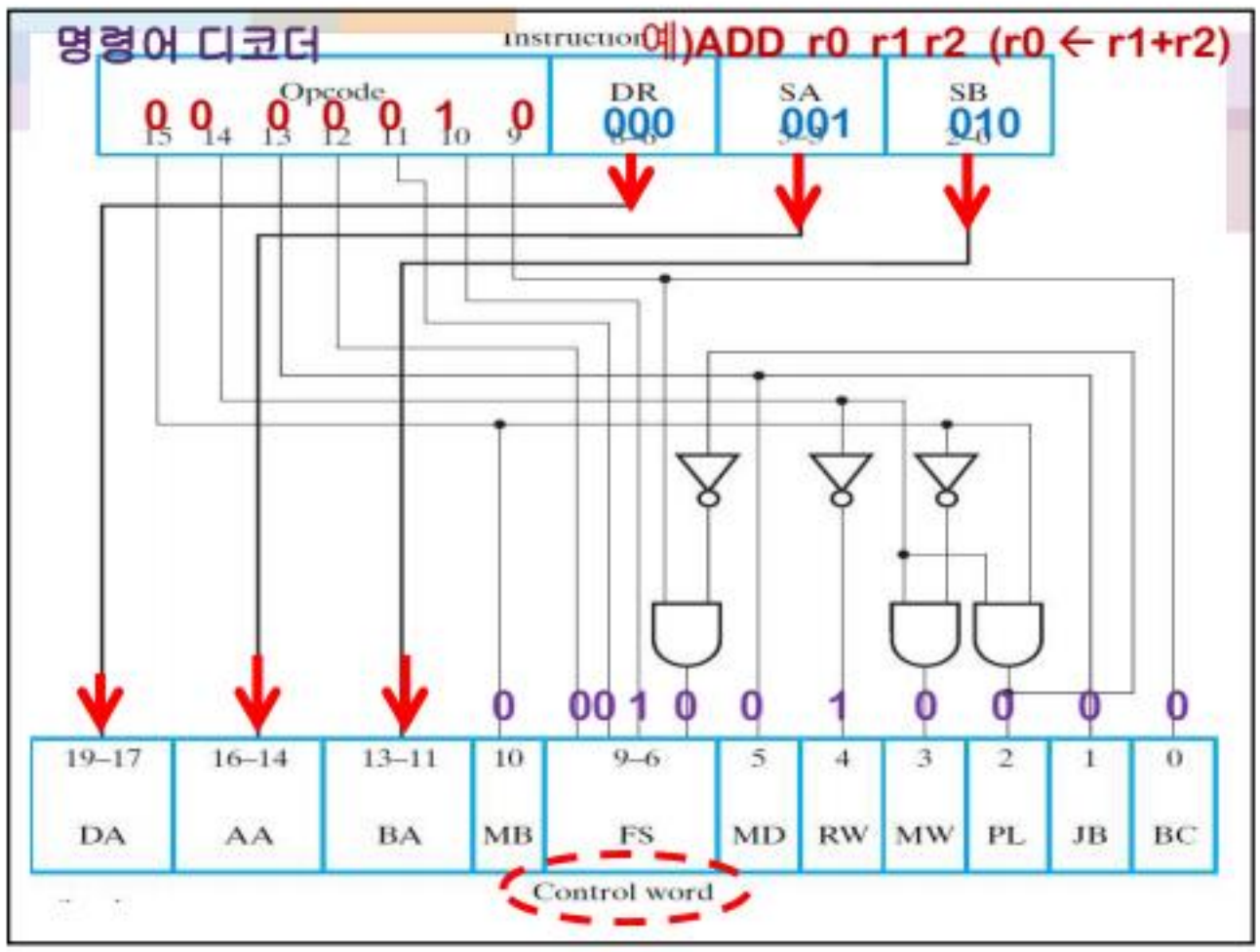
op15 op14 = 11일때 BRN에 예외발생, FS=0000어야 한다.

op15·op14

30

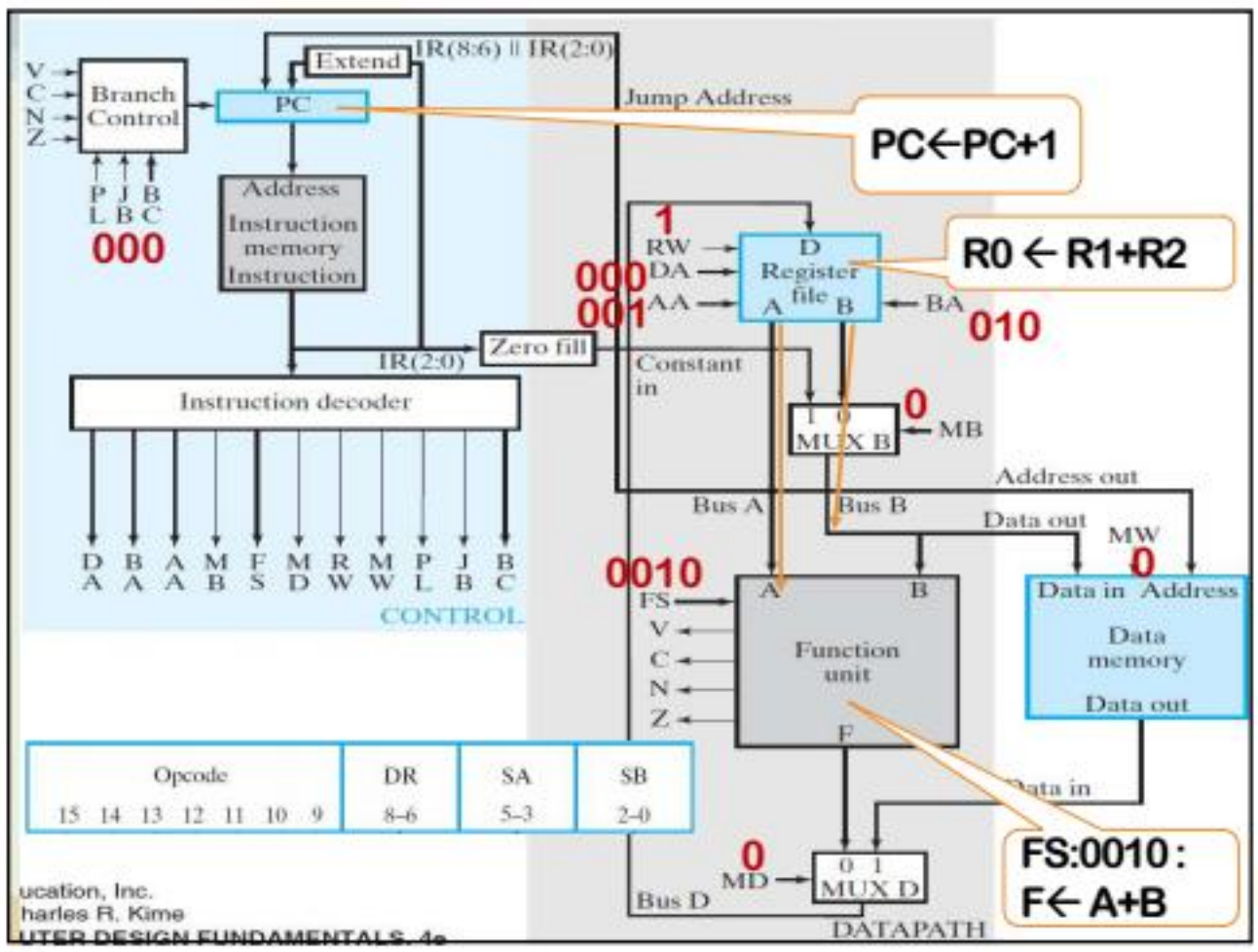

예)movA r0 r2 (r0 ← r2)
V
C
N
Z
Branch Control
PC
Extend
IR(8:6) || IR(2:0)
Jump Address
PC←PC+1
P L
J B
B C
000
Address
Instruction memory
Instruction
1
RW
DA
AA
000
010
D
Register file
A
B
BA
R0 ← R2
IR(2:0)
Zero fill
Constant in
Instruction decoder
1 0
MUX B
0
MB
Address out
Bus A
Bus B
Data out
MW
0
D A
B A
A A
M B
F S
M D
R W
M W
P L
J B
B C
CONTROL
0000
FS
V
C
N
Z
A
B
Function unit
F
Data in Address
Data memory
Data out
Opcode
15 14 13 12 11 10 9
DR
8–6
SA
5–3
SB
2–0
Data in
0
MD
0 1
MUX D
Bus D
FS:0000 :
F← A
DATAPATH
ucation, Inc.
harles R. Kime
UTER DESIGN FUNDAMENTALS, 4e

31


명령어 디코더
Instruction
예)ADD r0 r1 r2 (r0 ← r1+r2)
Opcode
0 0 0 0 0 1 0
15 14 13 12 11 10 9
DR
000
SA
001
SB
010
0 00 1 0 0 1 0 0 0 0
19–17
DA
16–14
AA
13–11
BA
10
MB
9–6
FS
5
MD
4
RW
3
MW
2
PL
1
JB
0
BC
Control word

32

16






## 명령어 함수 유형 1 P.448
## (레지스터 간 연산)

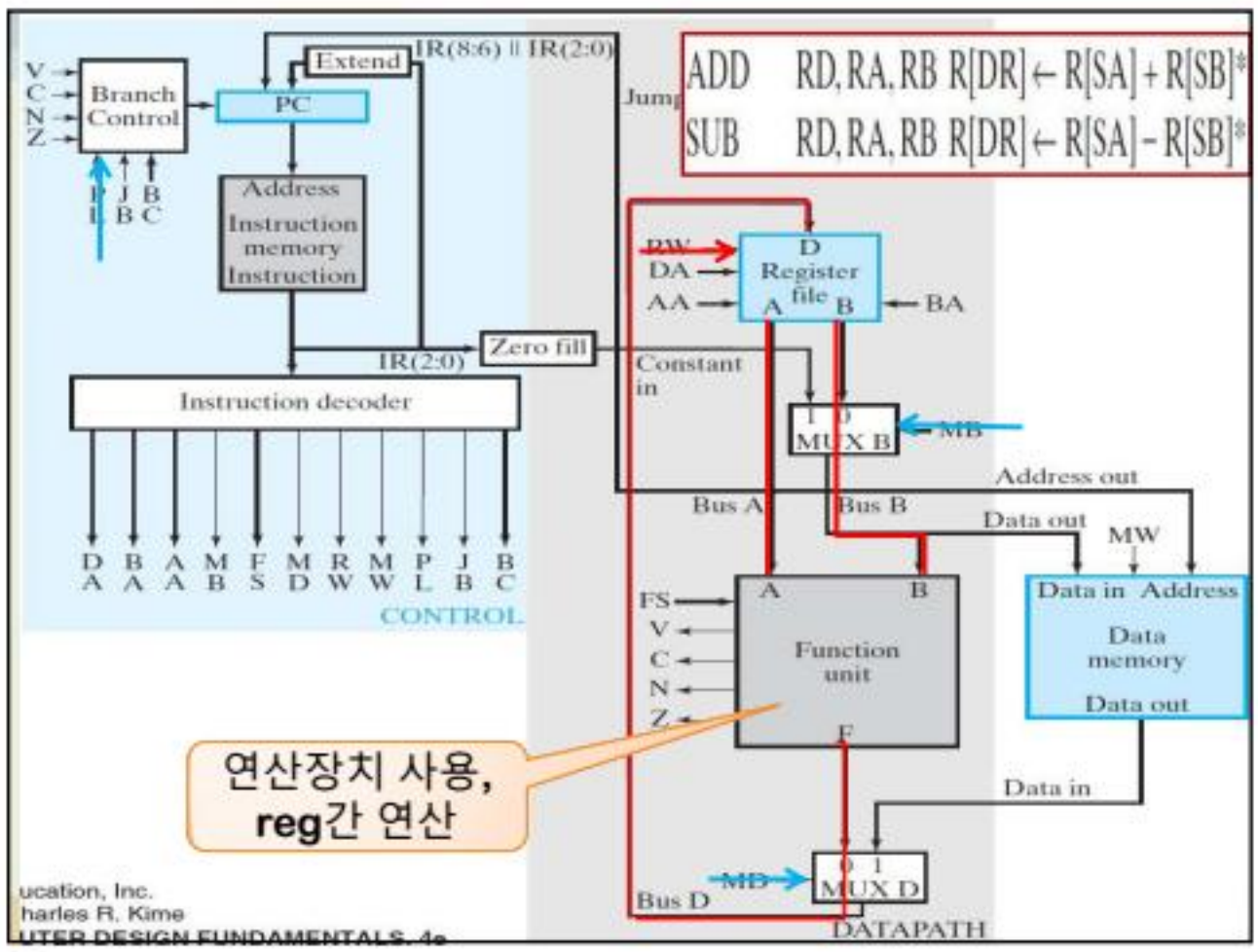
RegWrite

0: PC + 1
1: BR or JMP

| Instruction Function Type | Instruction Bits 15 | 14 | 13 | 9 | Control Word Bits MB | MD | RW | MW | PL | JB | BC |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Function-unit operations using registers | 0 | 0 | 0 | X | 0 | 0 | 1 | 0 | 0 | X | X |
| Memory read | 0 | 0 | | | | | | | | | |
| Memory write | 0 | 1 | | | | | | | | | |
| [illegible] | | | | | | | | | | | |
| Conditional branch on zero (Z) | 1 | 1 | | | | | | | | | |
| Conditional branch on negative (N) | 1 | 1 | | | | | | | | | |
| Unconditional jump | 1 | 1 | | | | | | | | | |

| | | | |
|---|---|---|---|
| Move A | 0000000 | MOVA | RD, RA |
| Increment | 0000001 | INC | RD, RA |
| Add | 0000010 | ADD | RD, RA, RB |
| Subtract | 0000101 | SUB | RD, RA, RB |
| Decrement | 0000110 | DEC | RD, RA |
| AND | 0001000 | AND | RD, RA, RB |
| OR | 0001001 | OR | RD, RA, RB |
| Exclusive OR | 0001010 | XOR | RD, RA, RB |
| NOT | 0001011 | NOT | RD, RA |
| Move B | 0001100 | MOVB | RD, RB |
| Shift Right | 0001101 | SHR | RD, RB |
| Shift Left | 0001110 | SHL | RD, RB |

**Branch control 논리에서**
**PL 이 0 이면**
**JB,BC은 상관없다.**




35

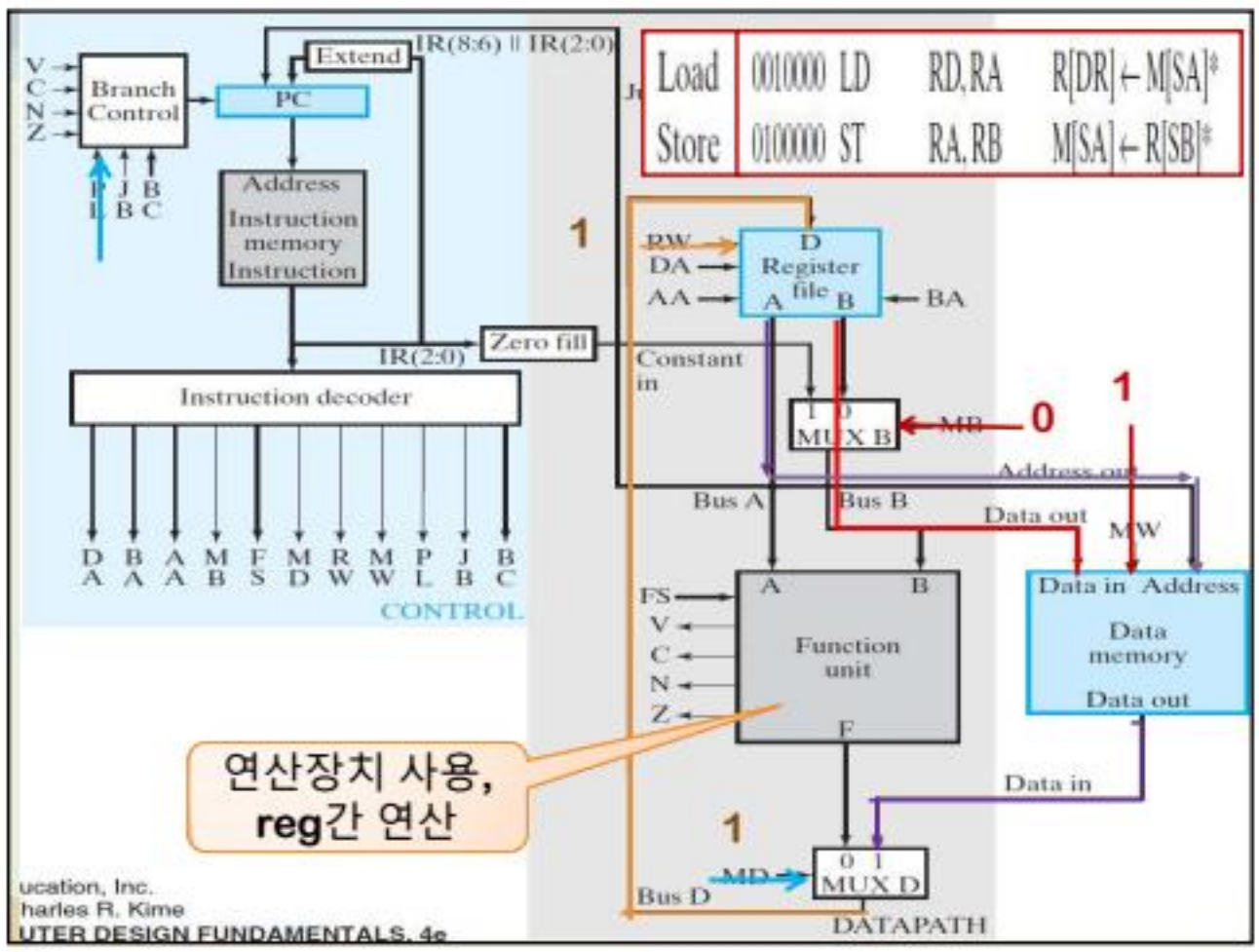
## 명령어 함수 유형 (메모리 읽기, 쓰기)

MemWrite

| Instruction Function Type | Instruction Bits 15 | 14 | 13 | 9 | Control Word Bits MB | MD | RW | MW | PL | JB | BC |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Function-unit operations using registers | 0 | 0 | 0 | X | 0 | 0 | 1 | 0 | 0 | X | X |
| Memory read | 0 | 0 | 1 | X | 0 | 1 | 1 | 0 | 0 | X | X |
| Memory write | 0 | 1 | 0 | X | 0 | X | 0 | 1 | 0 | X | X |
| Function-unit operations using register and constant | 1 | 0 | 0 | X | 1 | 0 | 1 | 0 | 0 | X | X |
| Con Load 0010000 LD RD, RA R[DR] ← M[SA]* | | | | | | | | | 1 | 0 | 0 |
| Con Store 0100000 ST RA, RB M[SA] ← R[SB]* | | | | | | | | | 1 | 0 | 1 |
| Unconditional jump | 1 | 1 | 1 | X | X | X | 0 | 0 | 1 | 1 | X |

18

36

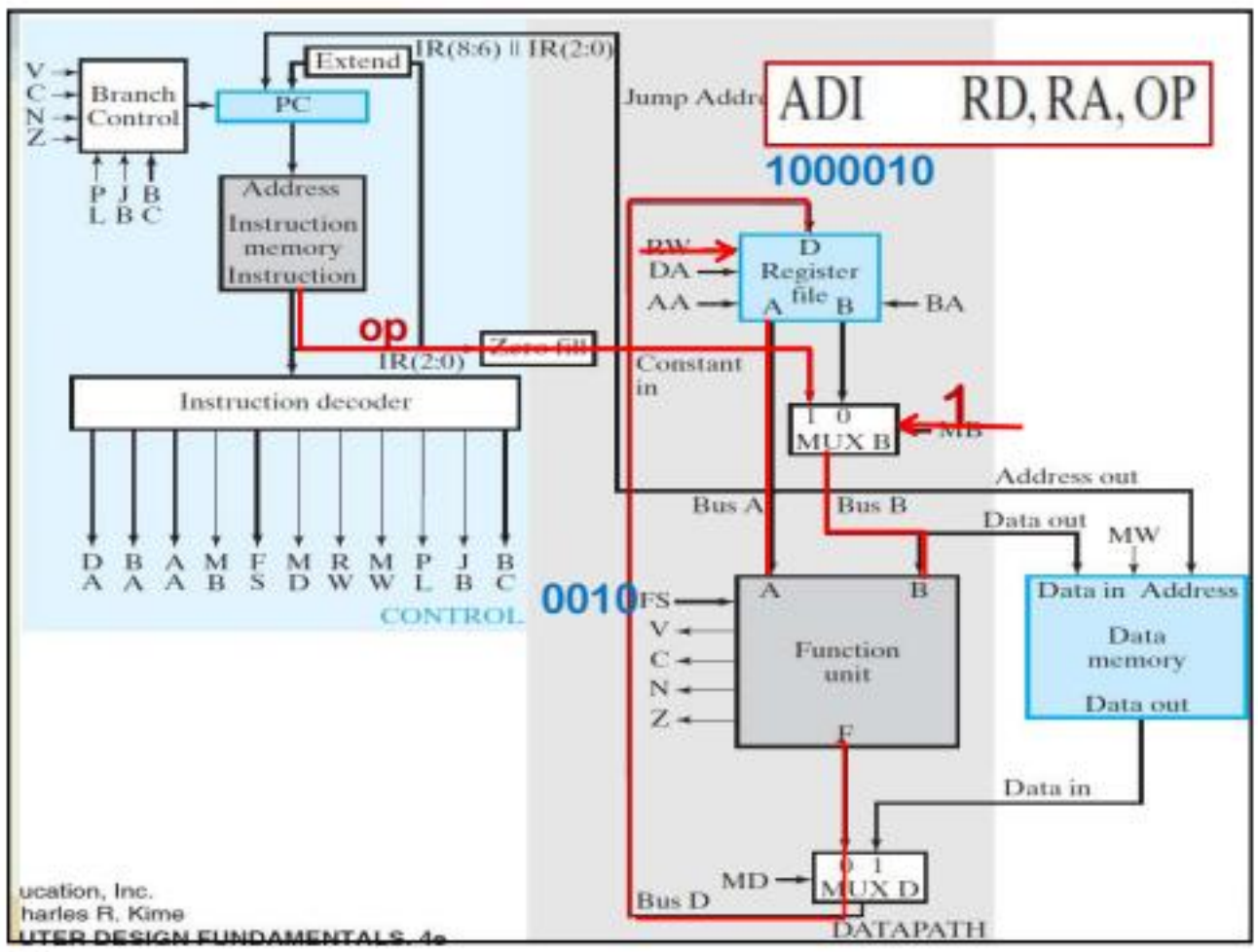



37

## 명령어 함수 유형 (reg와 상수간 연산)

| Instruction Function Type | Instruction Bits 15 | 14 | 13 | 9 | Control Word Bits MB | MD | RW | MW | PL | JB | BC |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Function-unit operations using registers | 0 | Load Immediate | | | 1001100 | LDI | | | RD, OP | | |
| | | Add Immediate | | | 1000010 | ADI | | | RD, RA, OP | | |
| Memory read | 0 | 0 | 1 | X | 0 | 1 | 1 | 0 | 0 | X | X |
| Memory write | 0 | 1 | 0 | X | 0 | X | 0 | 1 | 0 | X | X |
| Function-unit operations using register and constant | 1 | 0 | 0 | X | 1 | 0 | 1 | 0 | 0 | X | X |
| Conditional branch on zero ($Z$) | 1 | 1 | 0 | 0 | X | X | 0 | 0 | 1 | 0 | 0 |
| Conditional branch on negative ($N$) | 1 | 1 | 0 | 1 | X | X | 0 | 0 | 1 | 0 | 1 |
| Unconditional jump | 1 | 1 | 1 | X | X | X | 0 | 0 | 1 | 1 | X |

38




39

## 명령어 함수 유형 (조건분기, 점프)

조건분기
0: zero?
1: 음수?

**Branch control 제어입력**

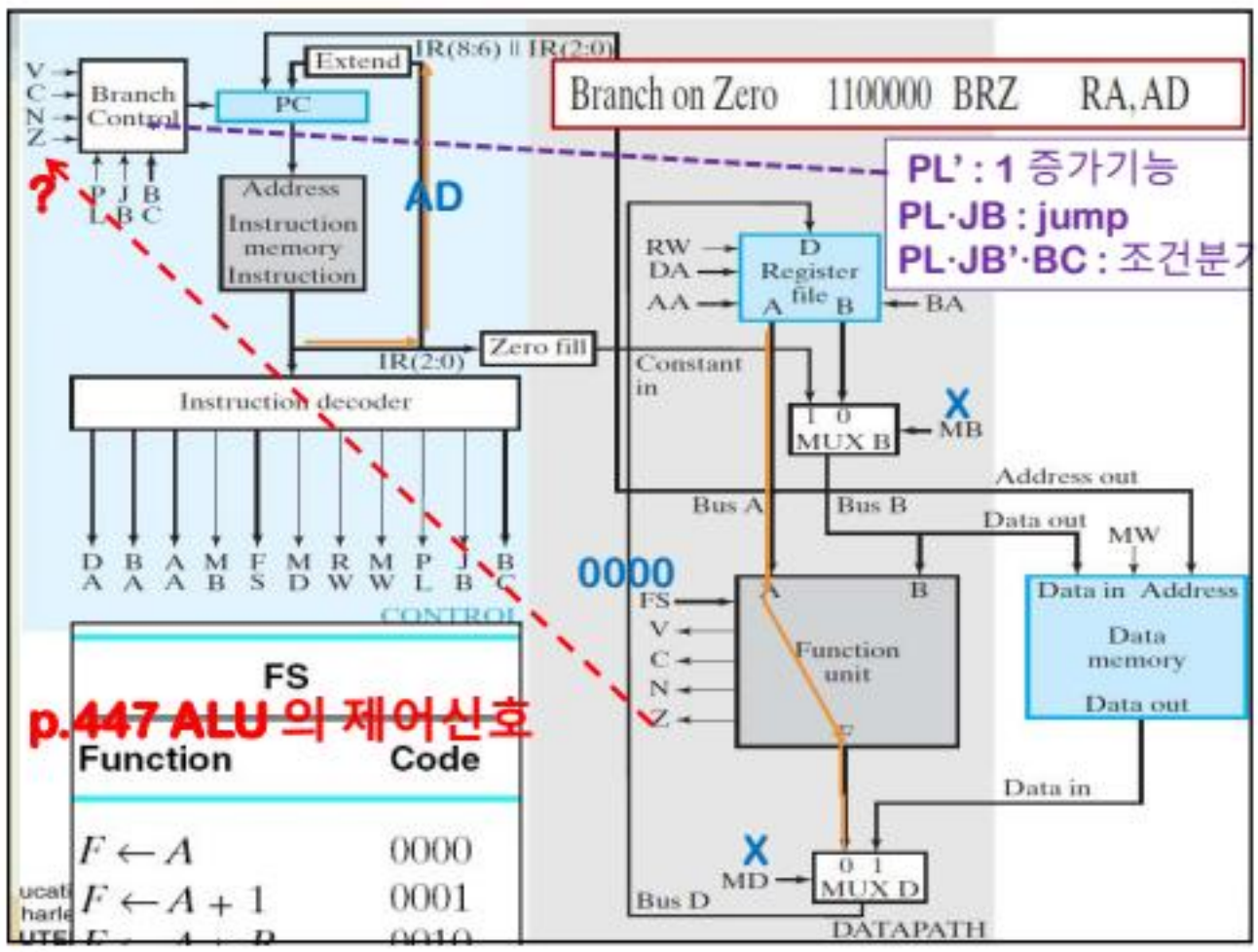
BRZ RA, AD

| Instruction Function Type | 15 | 14 | 13 | 9 | MB | MD | RW | MW | PL | JB | BC |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Function-unit operations using registers | 0 | Branch on Zero | | | 1100000 | BRZ | | | RA, AD | | |
| Memory read | 0 | Branch on | | | 1100001 | BRN | | | RA, AD | | |
| Memory write | 0 | Negative | | | | | | | | | |
| Function-unit operations using register and constant | 1 | Jump | | | 1110000 | JMP | | | RA | | |
| Conditional branch on zero (Z) | 1 | 1 | 0 | 0 | X | X | 0 | 0 | 1 | 0 | 0 |
| Conditional branch on negative (N) | 1 | 1 | 0 | 1 | X | X | 0 | 0 | 1 | 0 | 1 |
| Unconditional jump | 1 | 1 | 1 | X | X | X | 0 | 0 | 1 | 1 | X |

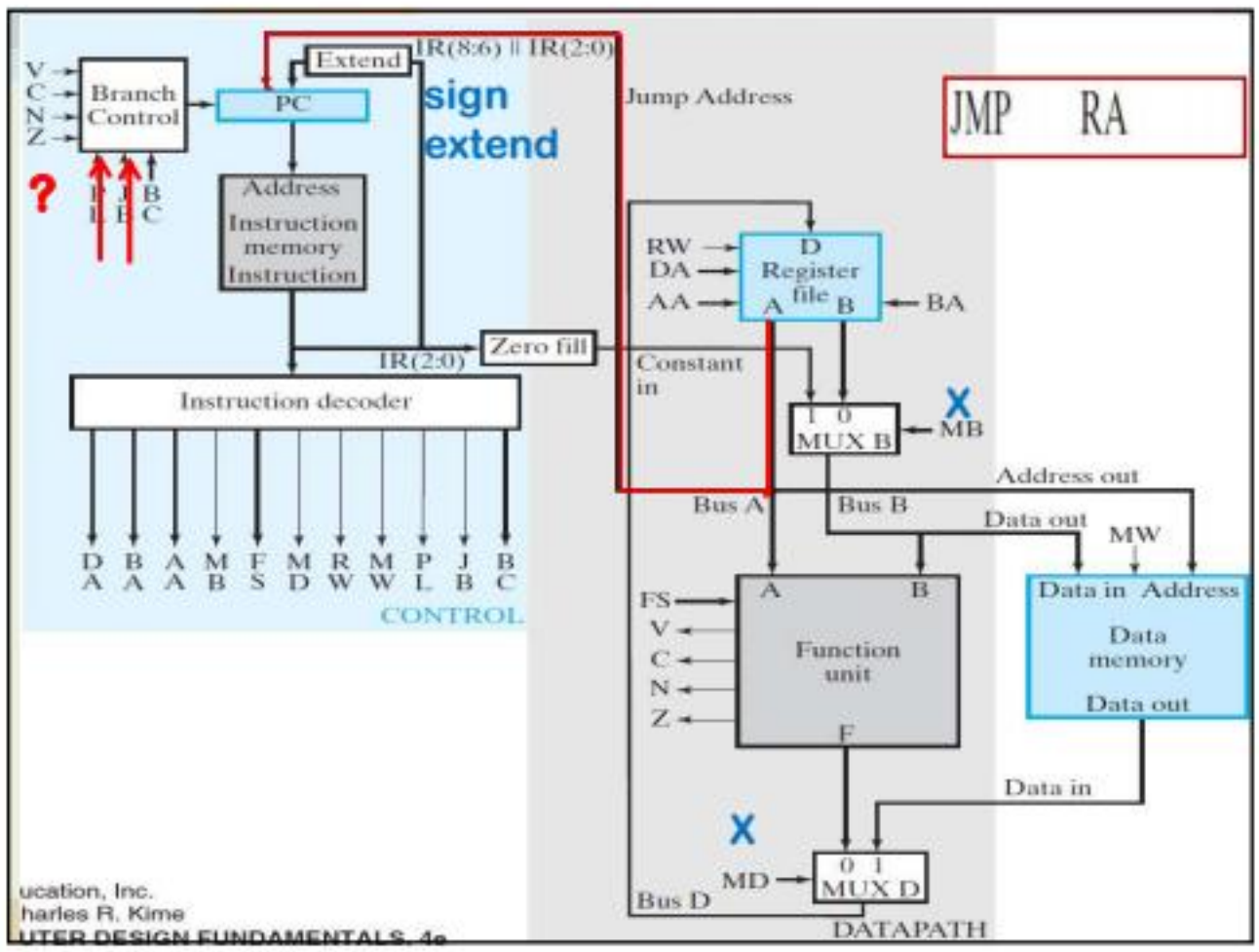
PL
0: PC + 1
1: BR or JMP

PL·JB: 무조건jump

40

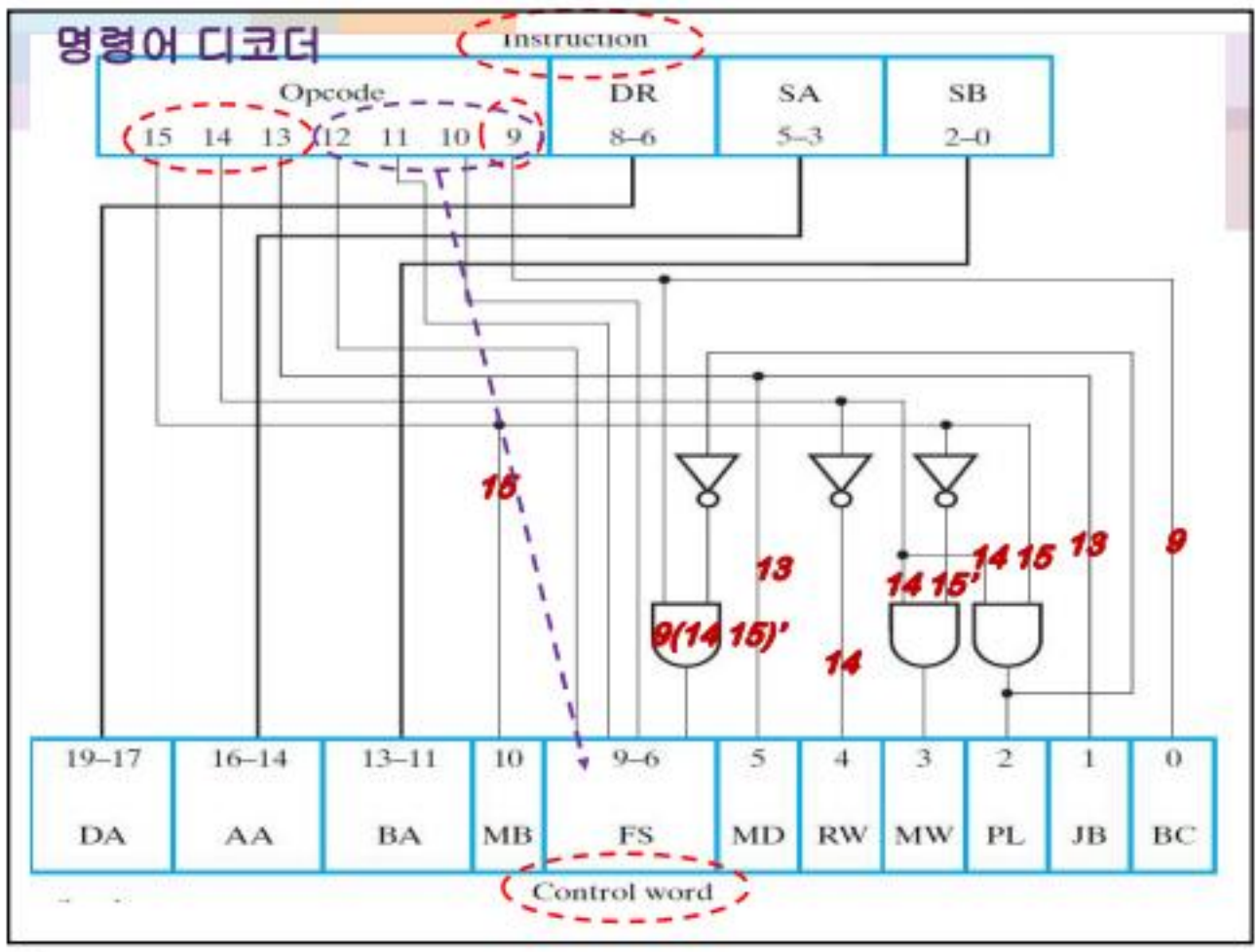

Branch on Zero 1100000 BRZ RA,AD
PL' : 1 증가기능
PL·JB : jump
PL·JB'·BC : 조건분기
Extend
IR(8:6) || IR(2:0)
V
C
N
Z
Branch Control
PC
Address
Instruction memory
Instruction
AD
IR(2:0)
Zero fill
Instruction decoder
RW
DA
AA
D
Register file
A
B
BA
Constant in
1 0
MUX B
MB
Address out
Bus A
Bus B
Data out
MW
D A B A A A M B F S M D R W M W P L J B B C
CONTROL
0000
FS
V
C
N
Z
Function unit
Data in Address
Data memory
Data out
Data in
MD
0 1
MUX D
Bus D
DATAPATH
FS
Function
Code
F ← A
0000
F ← A + 1
0001

41


Extend
IR(8:6) || IR(2:0)
sign extend
Jump Address
JMP RA
V
C
N
Z
Branch Control
PC
Address
Instruction memory
Instruction
IR(2:0)
Zero fill
Instruction decoder
RW
DA
AA
D
Register file
A
B
BA
Constant in
1 0
MUX B
MB
Address out
Bus A
Bus B
Data out
MW
D A B A A A M B F S M D R W M W P L J B B C
CONTROL
FS
V
C
N
Z
Function unit
Data in Address
Data memory
Data out
Data in
MD
0 1
MUX D
Bus D
DATAPATH

42

21




43

**if PL=0, FS=12,11,10,9**

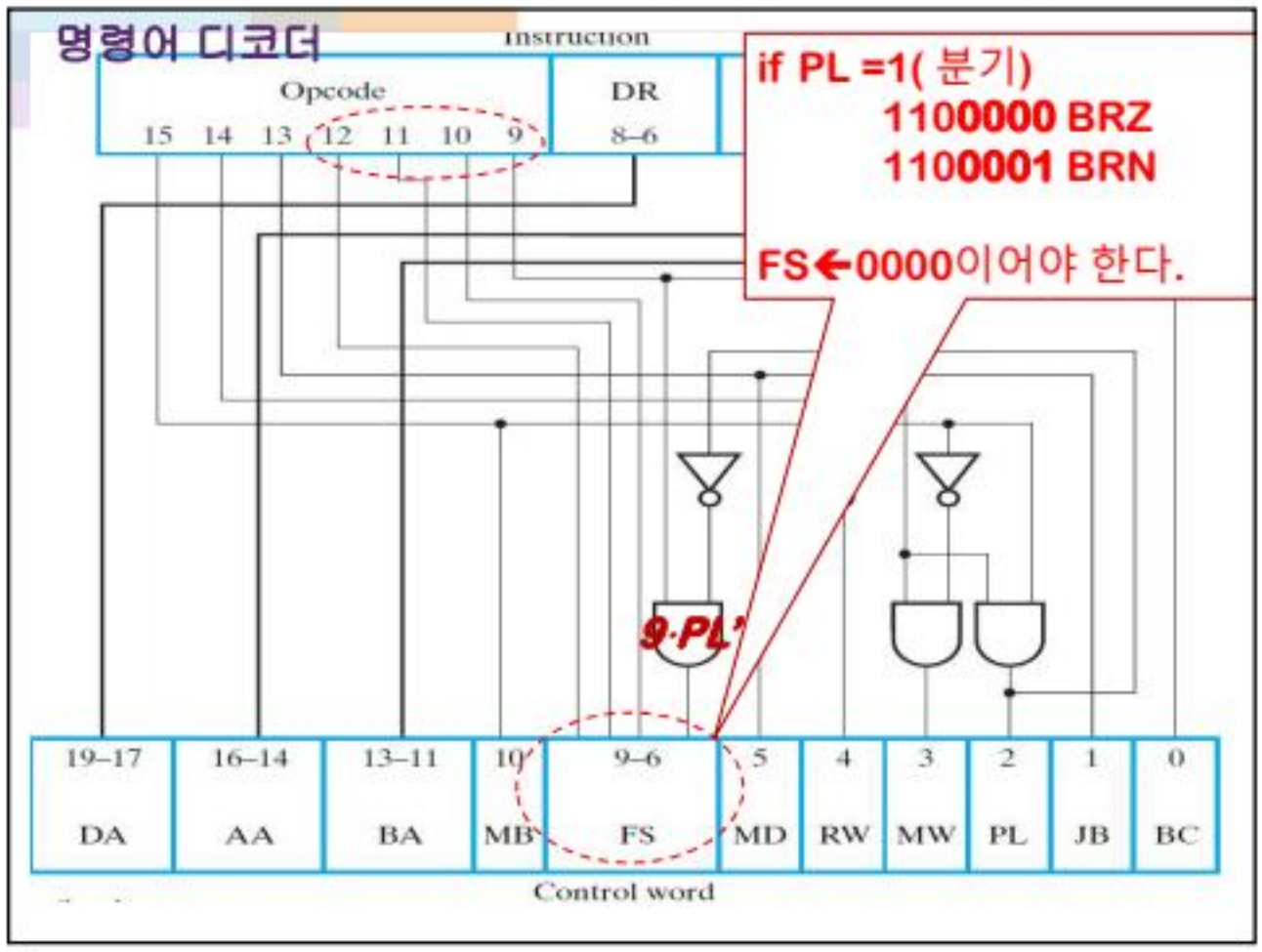
| Opcode | Mnemonic | Format | Description |
|---|---|---|---|
| 0000000 | MOVA | RD, RA | R[DR] ← R[SA]* |
| 0000001 | INC | RD, RA | R[DR] ← R[SA] + 1* |
| 0000010 | ADD | RD, RA, RB | R[DR] ← R[SA] + R[SB]* |
| 0000101 | SUB | RD, RA, RB | R[DR] ← R[SA] − R[SB]* |
| 0000110 | DEC | RD, RA | R[DR] ← R[SA] − 1* |
| 0001000 | AND | RD, RA, RB | R[DR] ← R[SA] ∧ R[SB]* |
| 0001001 | OR | RD, RA, RB | R[DR] ← R[SA] ∨ R[SB]* |
| 0001010 | XOR | RD, RA, RB | R[DR] ← R[SA] ⊕ R[SB]* |
| 0001011 | NOT | RD, RA | R[DR] ← $\overline{R[SA]}$ * |
| 0001100 | MOVB | RD, RB | R[DR] ← R[SB]* |
| 0001101 | SHR | RD, RB | R[DR] ← sr R[SB]* |
| 0001110 | SHL | RD, RB | R[DR] ← sl R[SB]* |

**ALU 의 제어신호 FS**

| Function | Code |
|---|---|
| $F \leftarrow A$ | 0000 |
| $F \leftarrow A + 1$ | 0001 |
| $F \leftarrow A + B$ | 0010 |
| $F \leftarrow A + B + 1$ | 0011 |
| $F \leftarrow A + \overline{B}$ | 0100 |
| $F \leftarrow A + \overline{B} + 1$ | 0101 |
| $F \leftarrow A - 1$ | 0110 |
| $F \leftarrow A$ | 0111 |
| $F \leftarrow A \wedge B$ | 1000 |
| $F \leftarrow A \vee B$ | 1001 |
| $F \leftarrow A \oplus B$ | 1010 |
| $F \leftarrow \overline{A}$ | 1011 |
| $F \leftarrow B$ | 1100 |
| $F \leftarrow \text{sr } B$ | 1101 |
| $F \leftarrow \text{sl } B$ | 1110 |

44

## 명령어 디코더




45

## 프로그램→명령어→제어신호 p.450

| | |
|---|---|
| LD R1, R3 | R1←M[R3] |
| ADI R1, R1, 3 | R1←R1+zf 3 |
| NOT R1, R1 | R1←R1' |
| INC R1, R1 | R1←R1+1 |
| INC R3, R3 | R3←R3+1 |
| LD R2, R3 | R2←M[R3] |
| ADD R2, R2, R1 | R2←R2+R1 |
| INC R3, R3 | R3←R3+1 |
| ST R3, R2 | M[R3]←R2 |

프로그램의 결과는?

가정)

| | |
|---|---|
| R0 | |
| R1 | |
| R2 | |
| R3 | 248 |
| | : |
| R7 | |

| Memory | |
|---|---|
| 248 | 2 |
| 249 | 83 |
| 250 | |

46

46

## 프로그램➔명령어➔제어신호 p.463

```
LD   R1, R3        R1←M[R3]
ADI  R1, R1, 3     R1←R1+zf 3
NOT R1, R1         R1←R1'
INC  R1, R1        R1←R1+1
INC  R3, R3        R3←R3+1
LD   R2, R3        R2←M[R3]
ADD R2, R2, R1     R2←R2+R1
INC  R3, R3        R3←R3+1
ST   R3, R2        M[R3]←R2
```

| | |
|---|---|
| R0 | |
| R1 | 2→5→5'→(5'+1) |
| R2 | 83→83+(5'+1) |
| R3 | 248→249→250 |
| | : |
| R7 | |

| Memory | |
|---|---|
| 248 | 2 |
| 249 | 83 |
| 250 | 83+(5'+1) |

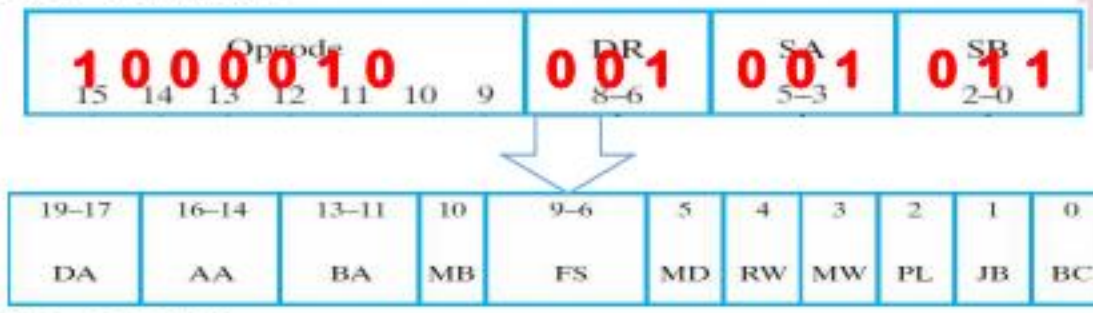
83+(-5)

47

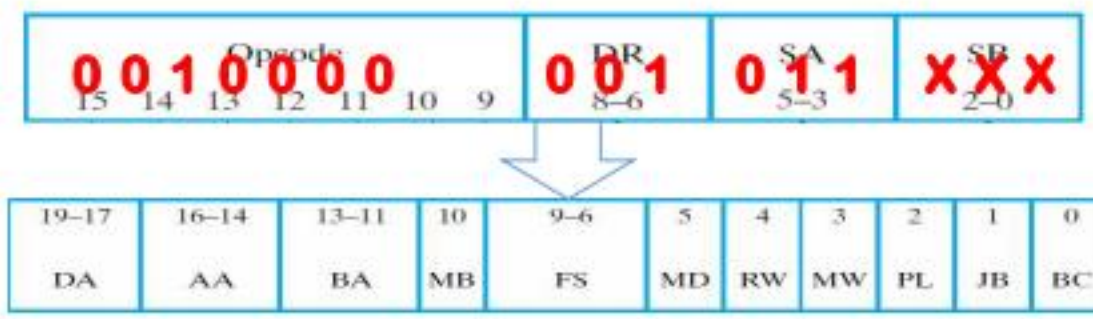
47

## 프로그램➔명령어➔제어신호 p.450

**ADI R1, R1, 3**




**LD R1, R3**




48

48

## 프로그램➔명령어➔제어신호

**ADI R1, R1, 3**

| Opcode (15 14 13 12 11 10 9) | DR (8–6) | SA (5–3) | SB (2–0) |
|---|---|---|---|
| 1 0 0 0 0 1 0 | 0 0 1 | 0 0 1 | 0 1 1 |

| 19–17 | 16–14 | 13–11 | 10 | 9–6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| DA | AA | BA | MB | FS | MD | RW | MW | PL | JB | BC |
| 001 | 001 | 011 | 1 | 0010 | 0 | 1 | 0 | 0 | 0 | 0 |

**LD R1, R3**

| Opcode (15 14 13 12 11 10 9) | DR (8–6) | SA (5–3) | SB (2–0) |
|---|---|---|---|
| 0 0 1 0 0 0 0 | 0 0 1 | 0 1 1 | X X X |

| 19–17 | 16–14 | 13–11 | 10 | 9–6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| DA | AA | BA | MB | FS | MD | RW | MW | PL | JB | BC |
| 001 | 011 | XXX | 0 | 0000 | 1 | 1 | 0 | 0 | 1 | 0 |

의미없음

49

49

## 프로그램➔명령어➔제어신호 p.461

확인해보자.

**ST R3, R2**

| Opcode (15 14 13 12 11 10 9) | DR (8–6) | SA (5–3) | SB (2–0) |
|---|---|---|---|

| 19–17 | 16–14 | 13–11 | 10 | 9–6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| DA | AA | BA | MB | FS | MD | RW | MW | PL | JB | BC |

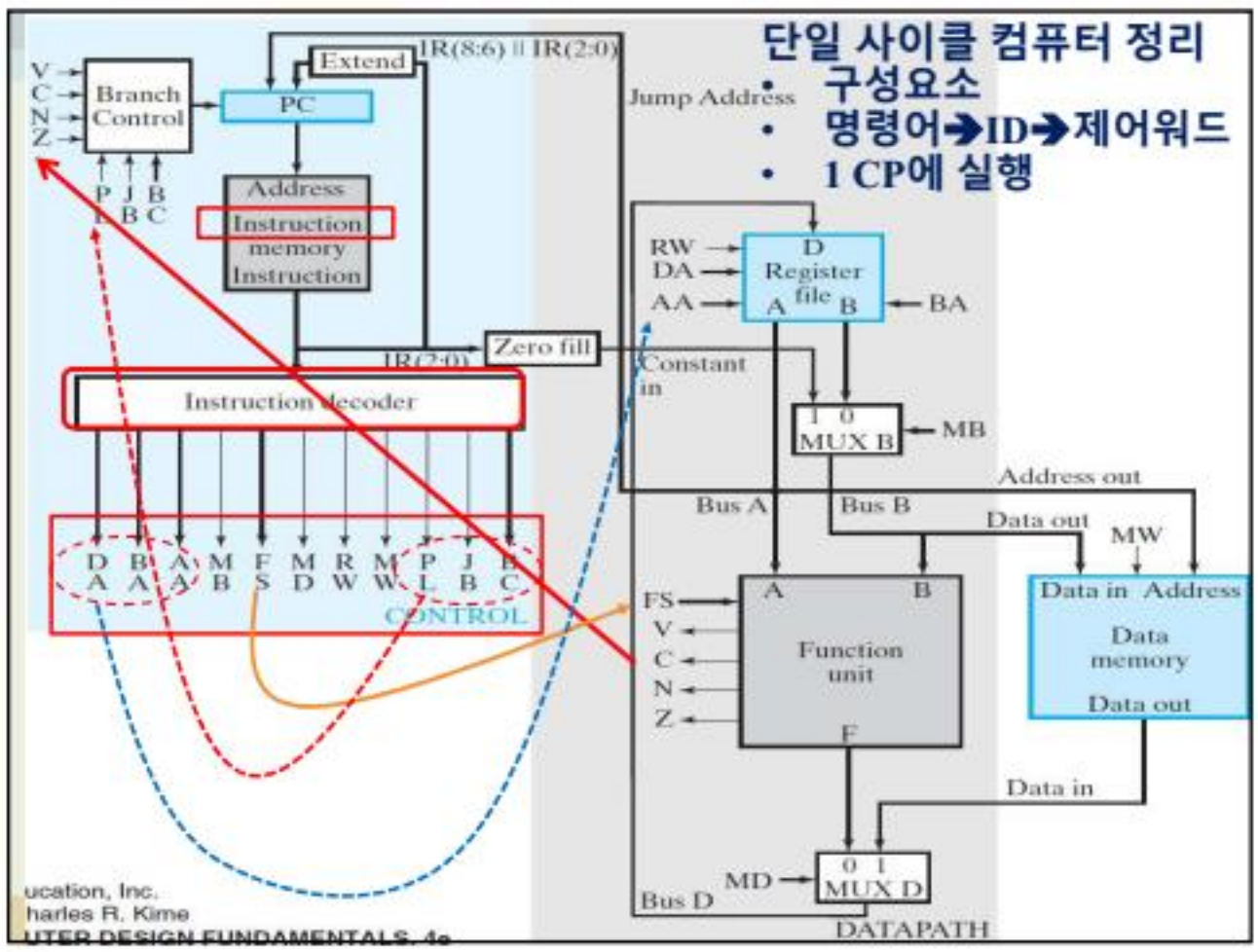
**BRZ R3, -10**

| Opcode (15 14 13 12 11 10 9) | DR (8–6) | SA (5–3) | SB (2–0) |
|---|---|---|---|

| 19–17 | 16–14 | 13–11 | 10 | 9–6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| DA | AA | BA | MB | FS | MD | RW | MW | PL | JB | BC |

50

50




51

## 정리

- single cycle computer
  - 명령어 형식
  - 명령어 유형 1,2,3,4
  - 제어장치의 이해

52