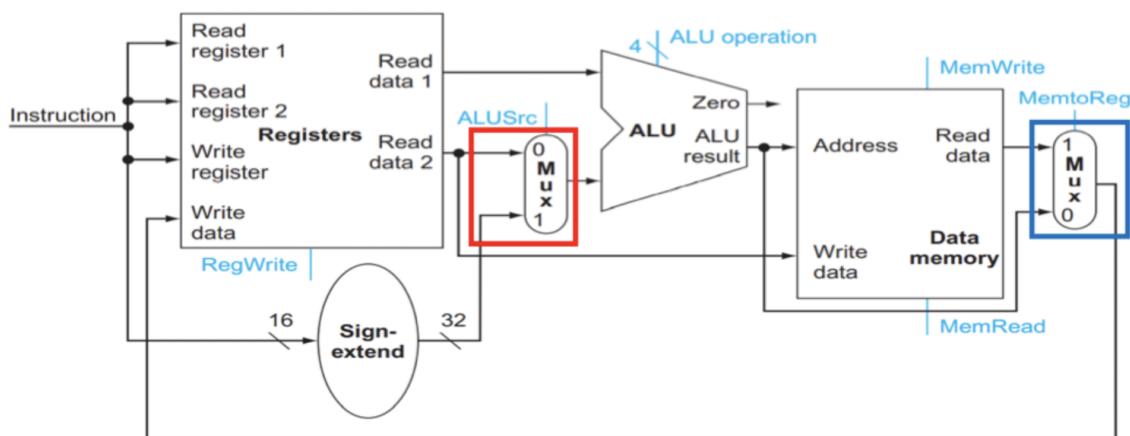


# Computer Architecture (ENE1004)

## ▼ Lec 9

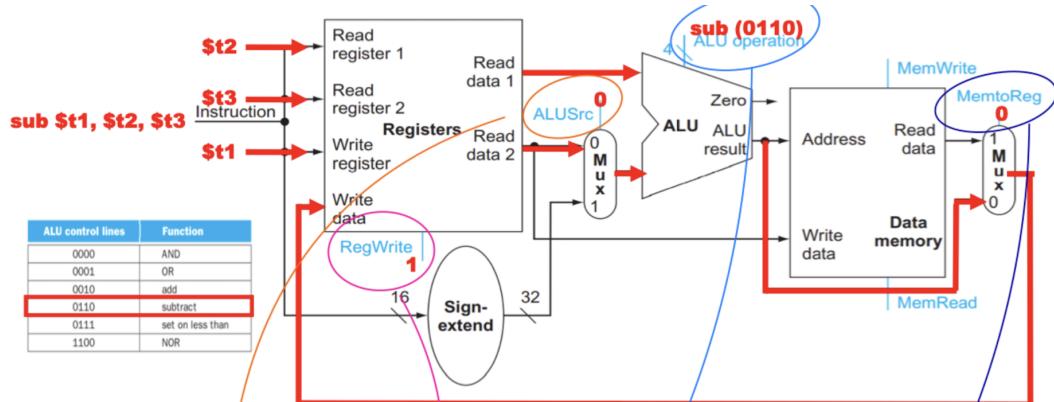
### Lec 9: The Processor 2

#### A Datapath for R-Type and Load/Store



- We combine (II) + (III) into a single datapath to execute any of the two types ((II) + (III))을 단일 데이터 경로로 결합하여 두가지 유형 중 하나를 실행한다.)
  - R-type (e.g., add \$t1, \$t2, \$t3): register file, ALU
  - Load/store (e.g., lw \$t1, 100(\$t2)): register file, ALU, sign extension unit(부호 확장 단위), data memory
- We add multiplexer (Mux) to select the required path for a given instruction (주어진 명령어에 필요한 경로를 선택하기 위해 multiplexer(Mux)을 추가한다.)
  - Mux for an input of the ALU: data from a register? or immediate from the instruction? (ALU의 입력을 위한 multiplexer: 레지스터의 데이터? 아니면 명령어에서 바로?)
  - Mux for the data input to the register file: data from the data memory? or result of the ALU? (레지스터 파일에 대한 데이터 입력을 위한 multiplexer: 데이터 메모리에서 데이터? 또는 ALU의 결과?)

## Execution of R-Type on the Datapath

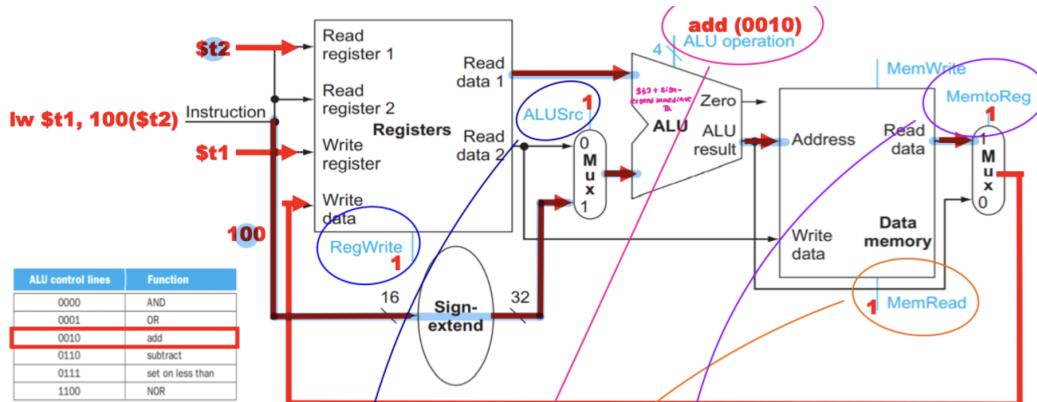


- **ALUSrc = 0** to select **Read data 2** of the register file to an input of ALU
- **ALU operation = 0110** to perform **sub** instruction on ALU
- **MemtoReg = 0** to select the data value of **ALU result** to supply **Write data** of the register file
- When writing into a register, **RegWrite = 1**

- ALUSrc = 0: ALU의 입력으로 레지스터 파일의 Read data 2를 선택한다.
- ALU operation = 0110: ALU에 하위 명령을 수행한다.
- MemtoReg = 0: 레지스터 파일의 쓰기 데이터를 제공할 ALU 결과의 데이터 값을 선택하기 위한 값이다.
- 레지스터에 쓰기 시, RegWrite = 1

1. 레지스터 파일(RF) 읽기 단계:
  - \$t2와 \$t3의 값이 레지스터 파일에서 읽힙니다.
  - 두 개의 값을 선택하기 위해 Mux가 사용됩니다. ALUSrc가 0이므로 이 Mux는 레지스터 파일의 읽기 데이터 2를 선택합니다.
2. ALU 연산 단계:
  - ALU는 선택된 두 입력 값을 사용하여 sub 연산을 수행합니다.
  - ALU에 sub 연산을 수행하도록 ALU Operation을 0110으로 설정합니다.
3. 레지스터 파일 쓰기 단계:
  - ALU에서 생성된 결과가 레지스터 파일로 쓰여집니다.
  - 이 때 Mux를 사용하여 레지스터 파일에 입력될 데이터를 선택합니다. MemtoReg가 0이므로 ALU의 결과가 레지스터 파일의 쓰기 데이터로 선택됩니다.
  - RegWrite가 1로 설정되어 있으므로, 레지스터 파일에 데이터가 쓰여집니다.

## Execution of Load Instruction on the Datapath



- **ALUSrc = 1** to select a sign-extended immediate value to an input of ALU
- **ALU operation = 0010** to perform add instruction on ALU
- **MemRead = 1** to read data from the data memory
- **MemtoReg = 1** to select the data read from the data memory to supply Write data of the register file
- When writing a register, **RegWrite = 1**

- ALUSrc = 1: ALU의 입력에 부호가 확장된 즉시 값을 선택함.
- ALU operation = 0010: ALU에 add 명령을 수행.
- MemRead = 1: 데이터 메모리에서 데이터를 읽음.
- MemtoReg = 1: 데이터 메모리에서 읽은 데이터를 선택하여 레지스터 파일의 쓰기 데이터를 제공
- 레지스터에 쓰기 시, Regwrite = 1

### 1. 레지스터 파일(RF) 읽기 단계:

- 레지스터 \$t2의 값을 레지스터 파일에서 읽어옵니다.
- 이 값은 주소 계산에 사용됩니다.
- ALUSrc가 1로 설정되어 있으므로, 레지스터에서 읽어온 값이 아닌 명령어에 포함된 sign-extended immediate 값이 ALU 입력으로 선택됩니다.

### 2. ALU 연산 단계:

- 선택된 레지스터 값(\$t2의 값)과 명령어에서 sign-extended immediate 값을 사용하여 ALU에서 덧셈(add) 연산을 수행합니다.
- ALU Operation을 0010으로 설정하여 덧셈(add) 연산을 수행합니다.

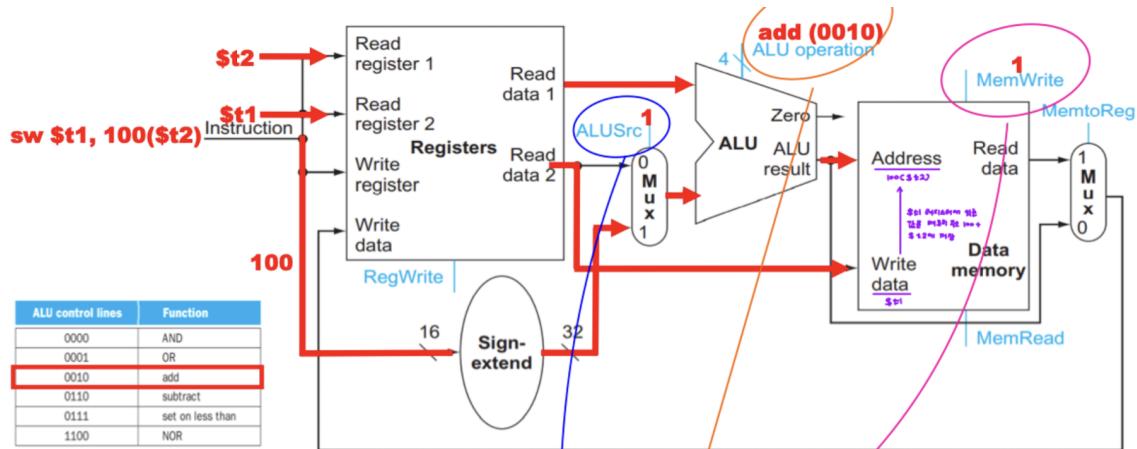
### 3. 데이터 메모리 읽기 단계:

- ALU에서 계산된 메모리 주소로부터 데이터를 데이터 메모리에서 읽어옵니다.
- MemRead가 1로 설정되어 있으므로, 데이터 메모리에서 데이터를 읽어옵니다.

### 4. 레지스터 파일 쓰기 단계:

- 읽어온 데이터를 레지스터 파일에 쓰기 위해 준비합니다.
- MemtoReg가 1로 설정되어 있으므로, 데이터 메모리에서 읽어온 데이터가 레지스터 파일에 쓰여질 데이터로 선택됩니다.
- RegWrite가 1로 설정되어 있으므로, 데이터가 레지스터 파일에 쓰여집니다.

## Execution of Store Instruction on the Datapath



- **ALUSrc = 1** to select a sign-extended immediate value to an input of ALU
- **ALU operation = 0010** to perform **add** instruction on ALU
- **MemWrite = 1** to write data into the data memory

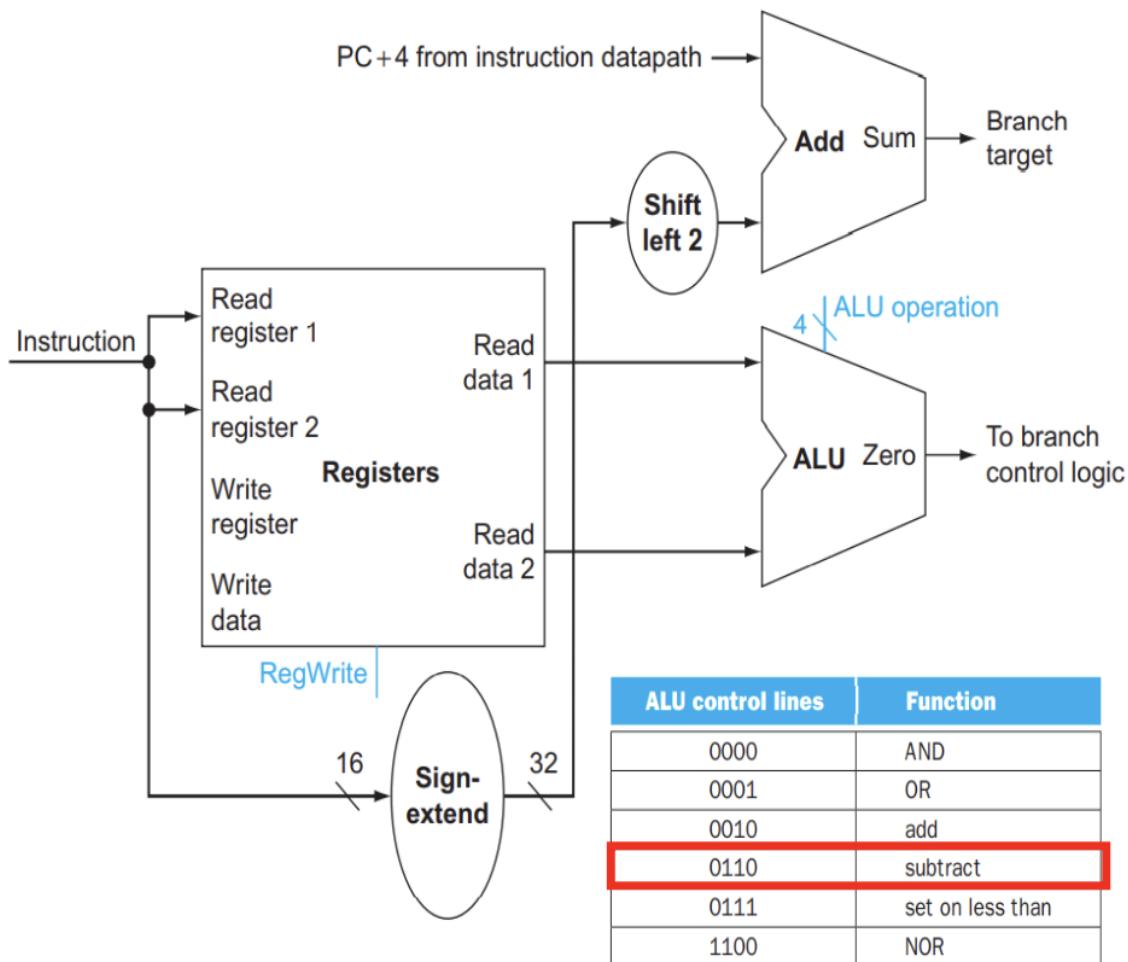
- ALUSrc = 1: ALU의 입력에 부호가 확장된 즉시 값을 선택함.
- ALU operation = 0010: ALU에 add 명령을 수행.
- MemWrite = 1: 데이터 메모리에 데이터를 씁.
- 1. 레지스터 파일(RF) 읽기 단계:
  - 레지스터 \$t1의 값을 레지스터 파일에서 읽어옵니다.
  - 이 값은 데이터 메모리에 저장할 데이터로 사용됩니다.
  - ALUSrc가 1로 설정되어 있으므로, 명령어에 포함된 sign-extended immediate 값을 ALU 입력으로 선택합니다.
- 2. ALU 연산 단계:
  - 선택된 레지스터 값(\$t2의 값)과 명령어에서 sign-extended immediate 값을 사용하여 ALU에서 덧셈(add) 연산을 수행합니다.
  - ALU Operation을 0010으로 설정하여 덧셈(add) 연산을 수행합니다.
- 3. 데이터 메모리 쓰기 단계:
  - ALU에서 계산된 메모리 주소로 데이터를 데이터 메모리에 쓰기 위해 준비합니다.
  - MemWrite가 1로 설정되어 있으므로, ALU에서 계산된 결과를 데이터 메모리에 씁니다.
  - \$t1 레지스터에 있는 값을 메모리 주소 100 + \$t2에 저장합니다.

## Datapath Elements for Branch (Branch에서 데이터 경로 요소)

- A branch instruction computes its target addresses using PC-relative addressing (branch 명령은 PC 상대 주소를 사용하여 대상 주소를 계산한다.)
  - beq \$t1, \$t2, offset\_value
  - Target address(대상 주소) = ① address of the subsequent instruction(후속 명령어의 주소) + ② branch offset in bytes
  - Target address = ① (PC + 4) + ② (offset\_value \* 4)

- Let us consider what datapath elements branch instructions require  
(branch 명령어에 필요한 데이터 경로 요소를 고려해보자.)
  - ①  $(PC + 4)$  can be obtained from the datapath for fetching an instruction ( $(PC+4)$ 는 명령어를 가져오기 위한 데이터 경로에서 얻을 수 있다.)
  - ②  $(offset\_value * 4)$  can be done by shifting left the offset\_value by 2 ( $(offset\_value * 4)$ 는  $offset\_value$ 를 2만큼 왼쪽으로 이동하여 얻을 수 있다.)
- Branch instructions can have two different scenarios depending on the conditions (Branch 명령어는 조건에 따라 두 가지 시나리오가 있을 수 있다.)
- (1) If it is true, the next instruction is the instruction at the target address  
(참인 경우, 다음 명령어는 대상 주소의 명령어이다.)
  - We say that the branch is "taken" (우리는 branch가 취해졌다고 말한다.)
  - $PC \leftarrow target\ address$
- (2) If it is not true, the next instruction is the instruction that follows sequentially (참이 아닌 경우, 다음 명령어는 순차적으로 이어지는 명령어이다.)
  - We say that the branch is "not taken" (우리는 branch가 취해지지 않았다고 말한다.)
  - $PC \leftarrow PC + 4$

## beq \$t1, \$t2, offset\_value

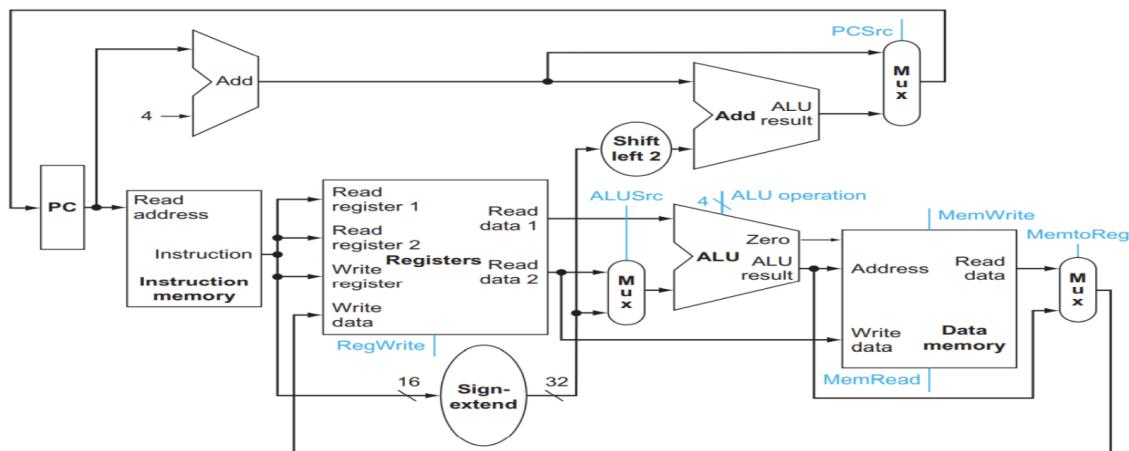


- (1) Computing the target address (대상 주소 계산)
  - Target address =  $(PC + 4) + (offset\_value * 4)$
  - **Sign extension unit(부호 확장 단위)** for  $(offset\_value)$
  - **Shift left 2 unit** for  $(offset\_value * 4)$
  - **Adder** for  $(PC + 4) + (offset\_value * 4)$
- (2) Comparing the register contents (레지스터 내용 비교)
  - **Register file** to supply two operands ( $\$t1 & \$t2$ ) : 두 연산자를 제공하기 위한 레지스터 파일이 필요함.
  - **ALU** for comparing the two operands ( $\$t1$  vs  $\$t2$ ) : 두 연산자를 비교하기 위해 ALU가 필요함.

- **ALU operation = 0110** for a subtract : 두 연산자의 차이를 구하기 위해 ALU 연산은 0110(subtract)로 설정함.
- If **Zero = 1** (two values are equal), the instruction in the target address should be executed next : 만약 zero 플래그가 1이면 (두 값이 동일하다면), 대상 주소에 있는 명령어가 실행됨.
- If **Zero = 0** (two values are not equal), the instruction that follows sequentially should be executed next : 만약 zero 플래그가 0이면 (두 값이 동일하지 않다면), 순차적으로 진행되어야 하는 명령어가 실행된다.

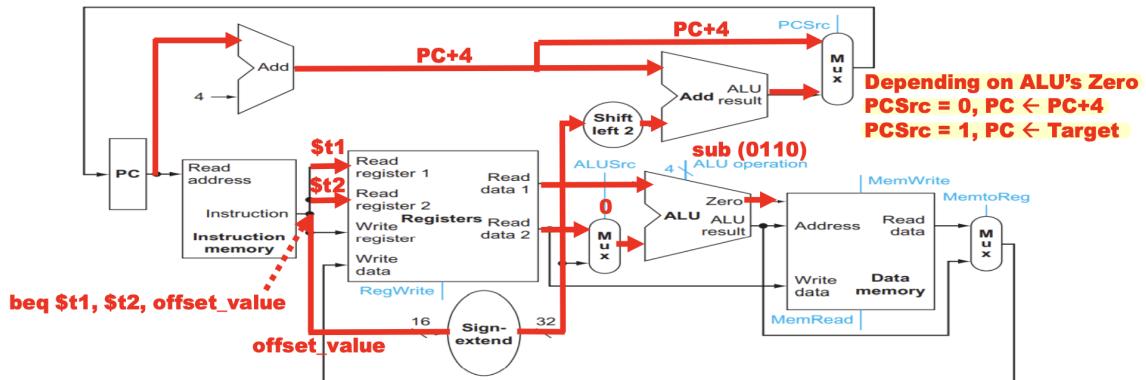
## A Single Datapath that Combines R, I, and Branch

단일 데이터 경로는 R, I, Branch를 모두 처리할 수 있도록 설계되어야 한다.



- For fetching instructions, we need PC, Instruction memory, Adder  
명령어 가져오기: PC, 명령어 메모리(Instruction memory), 덧셈기(Adder)가 필요함.
- For R-type instructions, we need Register file and ALU  
R 타입 명령어: 레지스터 파일과 ALU가 필요함.
- For load/store instructions, we need Register file, ALU, Sign extension unit, Data memory  
load/store 명령어: 레지스터 파일, ALU, 부호 확장 유닛(sign extension unit), 데이터 메모리가 필요함
- For branch instructions, we need Register file, ALU, Sign extension unit, Shift left 2 unit, Adder  
Branch 명령어: 레지스터 파일, ALU, 부호 확장 유닛(sign extension unit), 왼쪽으로 2비트 시프트하는 유닛(shift left 2 unit), 덧셈기(Adder)가 필요함.

## Execution of Branch Instruction on the Datapath



- (1) Computing the target address:  $(PC+4) + (offset\_value * 4)$  using sign extension / shift left 2 / adder
- (2) At the same time, comparing the two register operands: **(\$t1 vs \$t2)** using register file / ALU
- The result of (2) is evaluated whether it is zero or not using **ALU's Zero**
- Based on the evaluation, either  $(PC+4)$  or  $\{(PC+4)+(offset\_value*4)\}$  is selected using **PCSrc = 0 / 1**