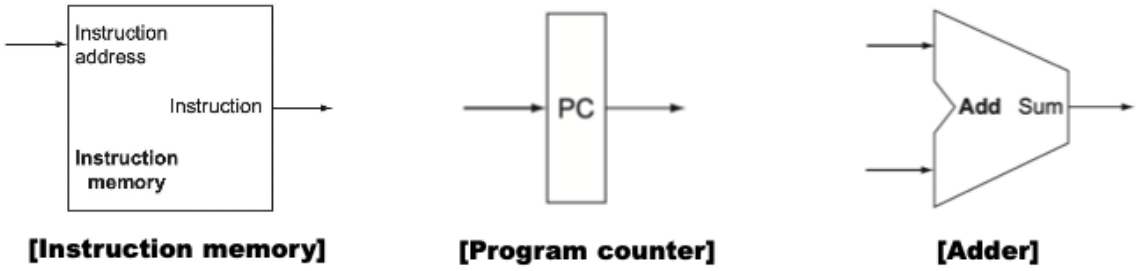
Computer Architecture (ENE1004)

▼ Lec 8

Lec 8: The Processor 1

(I) Datapath Elements for an Instuction (명령어의 데이터 경로 요소)

First, processors should know which instruction they will execute now (먼저, 프로세스는 지금 실행할 명령어를 알아야한다.)

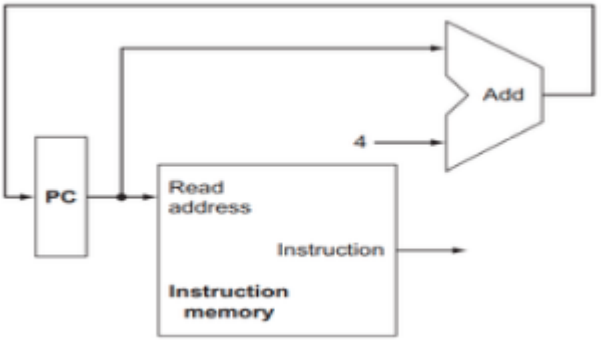


[Instruction memory] [Program counter] [Adder]

- Instruction memory: a memory unit to store the instructions of a program
  - Given an address (input), it supplies the corresponding instruction (output)
  - This is actually part of the memory (text segment)
- Program counter (PC): a register that holds the address of the current instruction
- Adder: a logic that increments the PC for the address of the next instruction
- Instruction memory: 프로그램의 명령어를 저장하는 메모리 단위
  - 주소(입력)가 주어지면, 해당 명령어(출력)을 제공한다.
  - 이것은 실제로 메모리의 일부(텍스트 세그먼트)이다.
- Program counter (PC): 현재 명령어의 주소를 보관하는 레지스터

- Adder: 다음 명령어의 주소를 위해 PC를 증가시키는 논리

A Datapath for Fetching Instructions (명령어 불러오기를 위한 데이터 경로)



- (1) To execute an instruction, we must “fetch” the instruction from memory (명령어를 실행하려면 메모리에서 명령어를 불러와야(fetch) 한다.)
  - The processor obtains an instruction when the address of PC is given to the instruction memory (프로세서는 명령어 메모리에 PC의 주소가 주어지면 명령어를 가져온다.)
- (2) To prepare for the execution of the next instruction, we must also “update (increment) the PC” so that if points to the next (sequential) instruction (다음 명령어의 실행을 준비하기 위해 다음 (순차적) 명령을 가리키도록 PC를 업데이트(증가) 해야한다.)
  - The size of an instruction is 4 bytes (명령어의 크기는 4바이트이다.)
  - For jump and branch instructions, PC will point at the target address (instruction) (jump와 branch 명령어의 경우, PC는 대상 주소(명령어)를 가르킨다.)

(II) Datapath Elements for R-Type (R-Type의 데이터 경로 요소)

[Register file]

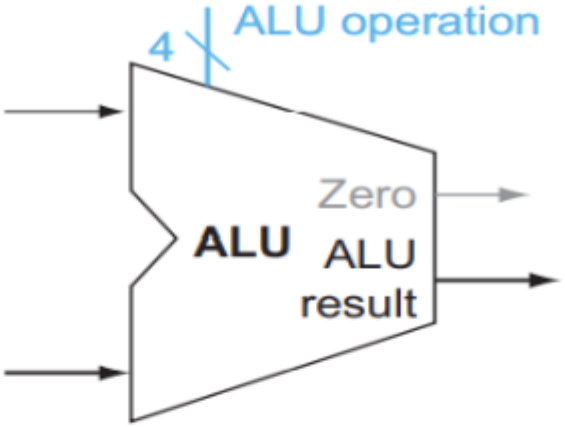
* Source 1 register (read):
Input (Read register 1) → Output (Read data 1)

* Source 2 register (read):
Input (Read register 2) → Output (Read data 2)

* Destination register (write):
Inputs (Write register & Write data)
& signal (RegWrite) = 1 → No output

- R-type instructions read two registers, perform an ALU operation on the contents of the registers, and write the result to a register (R타입 명령어는 두 개의 레지스터를 읽고, 레지스터의 내용에 대해 ALU 연산을 수행한 후 그 결과를 레지스터에 기록한다.)
  - E.g., add, sub, and, or, slt
- Register file: a collection of the processor's 32 general-purpose registers (레지스터 파일: 프로시저의 32개 범용 레지스터 모음)
  - For two data words to be read, it provides two inputs that specify the register numbers (Read registers 1 & 2) and two outputs that carry the values of the registers (Read data 1 & 2)

- To write a data word, it provides two inputs: one to specify the register number to be written (Write register) and one to supply the data to be written (Write data)
- A write is controlled by a control signal (RegWrite), which must be asserted for a write, while reads are always serviced on the Read register inputs
- Register inputs are 5 bits wide; data input/output buses are each 32 bits wide
- 두 개의 데이터 워드를 읽기 위해 레지스터 번호를 저장하는 두 개의 입력 (read register1 및 read register2)을 제공한다.
- 데이터 워드를 쓰기 위해서는 기록할 레지스터 번호를 저장하는 입력(write register)과 기록할 데이터를 공급하는 입력(write data) 두 개를 제공한다.
- 쓰기는 쓰기를 위해 주장되어야 하는 제어 신호(Regwrite)에 의해 제어되며, 읽기는 항상 읽기 레지스터 입력에서 제공된다.
- register 입력은 5비트 폭이며, 데이터 입력/출력 버스는 각각 32비트 폭이다.



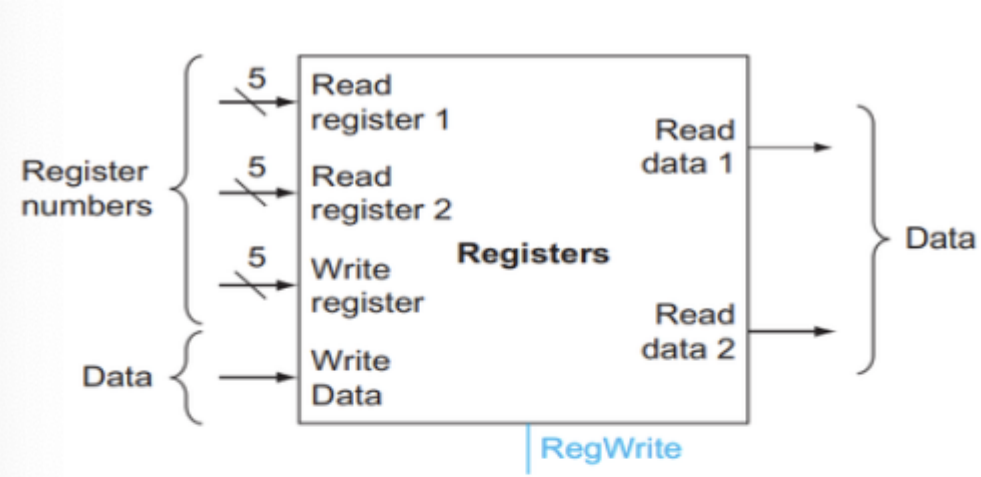
[Arithmetic Logic Unit (ALU)]

ALU control lines	Function
0000	AND
0001	OR
0010	add
0110	subtract
0111	set on less than
1100	NOR

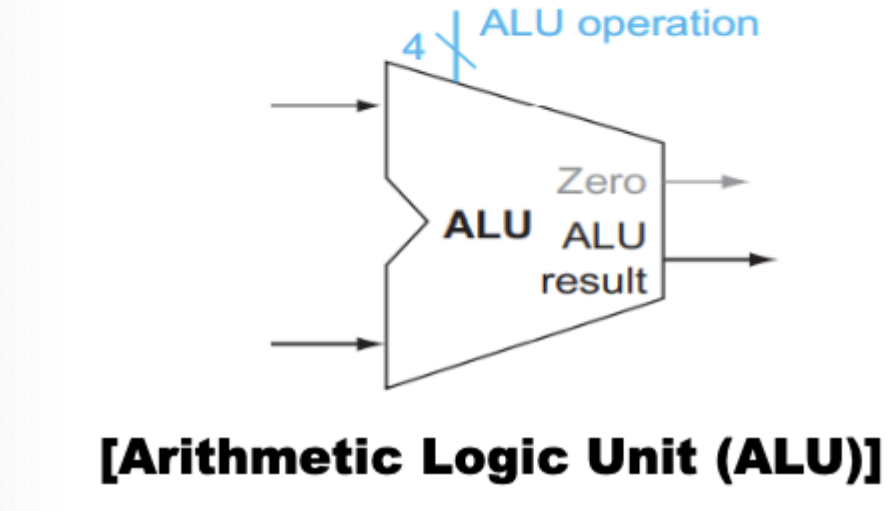
- ALU: a unit that can perform various arithmetic, logical operations on given inputs (주어진 입력에 대한 다양한 산술적, 논리적 연산을 수행할 수 있는 단위)
  - It takes two 32-bit operands as inputs (2개의 32비트 피연산자를 입력으로 받는다.)
  - It produces a 32-bit result (ALU result) as an output (32비트 결과 (ALU 결과)를 출력으로 생성한다.)
  - It produces a 1-bit signal (Zero) if the result is 0 (결과가 0이면 1 비트 신호 (0)을 출력한다.)
  - The 4-bit control signal (ALU operation) determines what operation the ALU performs (4비트 제어 신호(ALU 연산)는 ALU가 수행하는 연산을

결정한다.)

(III) Dathpath Elements for Load/Store (Load/Store을 위한 데이터 경로 요소)



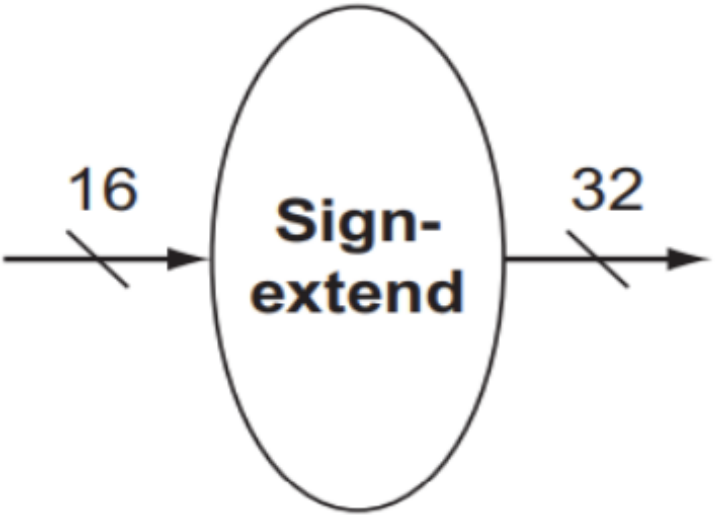
[Register file]



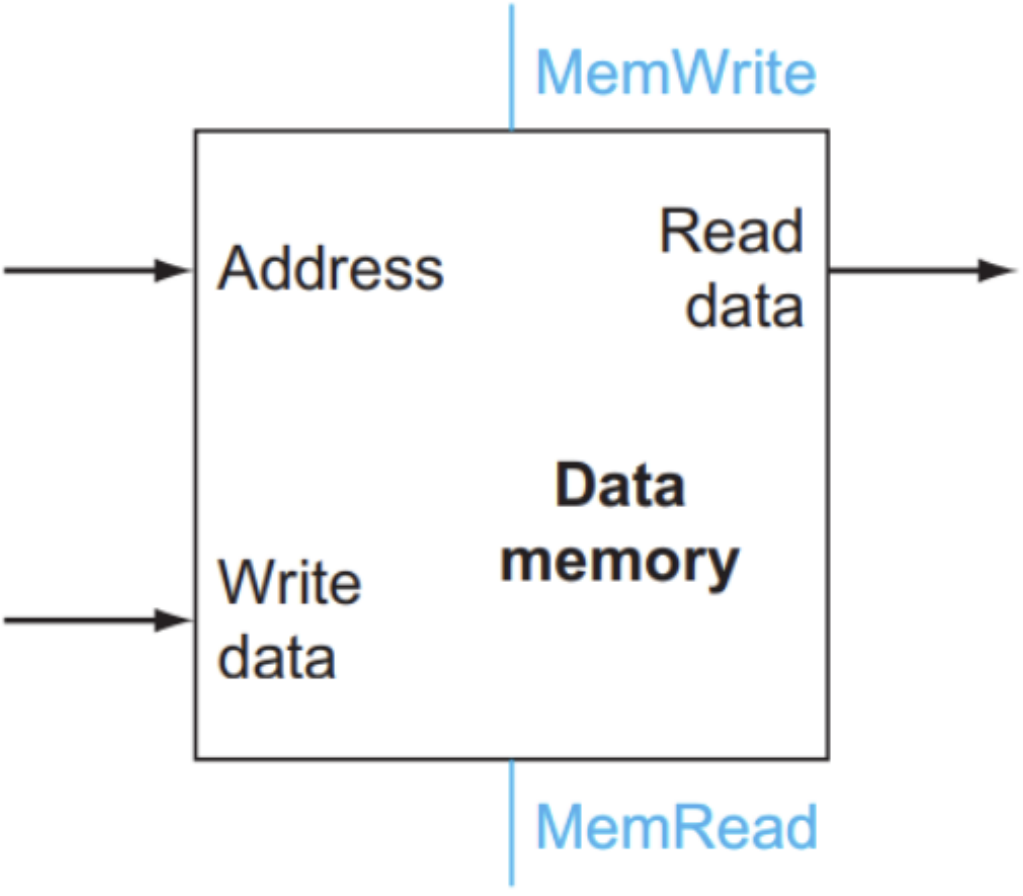
[Arithmetic Logic Unit (ALU)]

- A load/store instruction computes a memory address by adding the base register to the 16-bit signed offset field contained in the instructions (로드/저장 명령어는 명령어에 포함된 16비트 부호화 된 오프셋 필드에 베이스 레지스터를 추가하여 메모리 주소를 계산한다.)
  - E.g., lw $t1, offset_value($t2)
  - E.g., sw $t1, offset_value($t2)
  - The value of $t2 is read from the register file ($t2의 값은 레지스터 파일에서 읽는다.)

    - $t2 + offset_value can be done by the ALU ($t2 + offset_value는 ALU에서 수행할 수 있다.)
- For a load instruction(lw), the value read from memory must be written into the register file ($t1) (로드 명령(lw)의 경우, 메모리에서 읽은 값을 레지스터 파일($t1)에 써야한다.)
    - Write register, write data, & write signal should be set (쓰기 레지스터, 쓰기 데이터, 쓰기 신호가 설정되어야 한다.)
- For a store instruction (sw),the value to be stored must be read from the register file ($t1) (저장 명령(sw)의 경우, 저장할 값을 레지스터 파일($t1)에서 읽어야 한다.)
    - Note that $t2 is read for computing the memory address (메모리 주소를 계산하기 위해 $t2를 읽는다.)



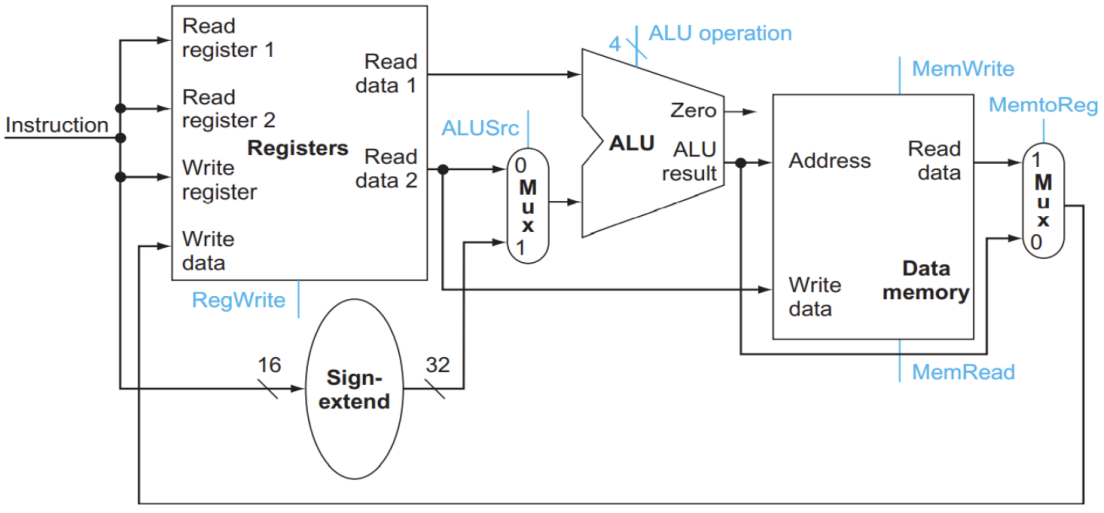
[Sign extension unit]



[Data memory]

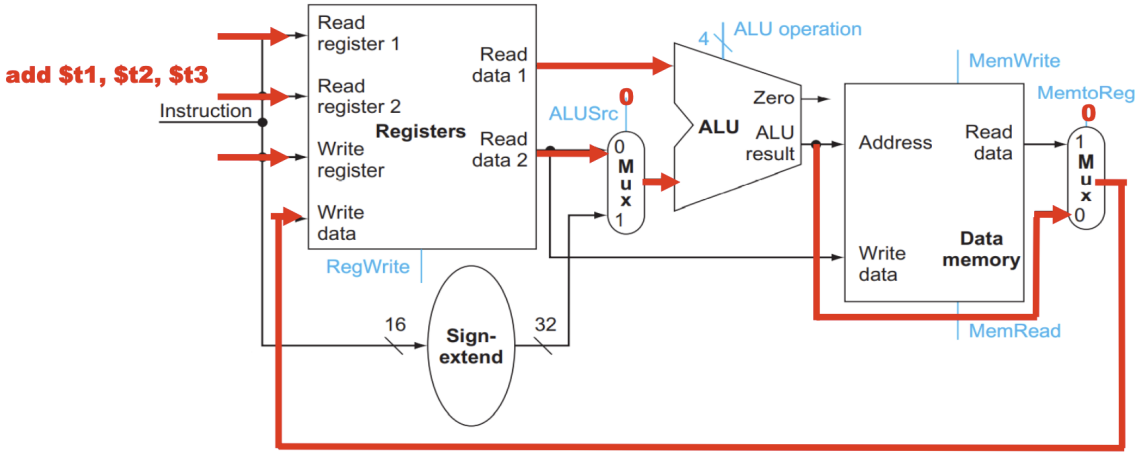
- In addition, we need a unit to sign-extend the 16-bit offset field in the instruction to a 32-bit signed value (명령어의 16비트 오프셋 필드를 32비트 부호화 된 값으로 부호 확장하는 단위가 필요하다.)
    - lw $t1, **offset_value**($t2) / sw $t1, **offset_value**($t2)
    - Sign extension unit (부호 확장 단위): takes a 16-bit signed value as an input and produces a 32-bit sign-extended value as an output (16비트 부호 값을 입력으로 받고 32비트 부호 확장 값을 출력으로 생성한다.)
    - **0**XXXXXXXXXXXXXXX → **0000000000000000** **0**XXXXXXXXXXXXXXX
    - **1**XXXXXXXXXXXXXXX → **1111111111111111** **1**XXXXXXXXXXXXXXX
- Data memory: a mem unit where data is read/stored (데이터를 읽기/저장하는 메모리 단위)
    - It offers seperate control signals (MemWrite and MemRead), each of which is asserted for a write and a read, respectively (쓰기와 읽기에 대해 각각 주장하는 별도의 제어신호 (MemWrite and MemRead)를 제공한다.)
    - For a memory read (lw), the computed address is given (Address) and the corresponding data is provided (Read data) (메모리 읽기(lw)의 경우, 계산된 주소가 제공되고(주소) 해당 데이터가 제공된다.(읽기 데이터))
    - For a memory write (sw), the data to be written is given (Write data) while the computed address is given well (메모리 쓰기(sw)의 경우, 쓰기할 데이터가 주어지고(쓰기 데이터), 계산된 주소도 함께 제공된다.)

A Datapath for R-Type and Load/Store (II) + (III)



- 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, 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.)

Execution of R-Type on the Datapath



- ALUSrc = 0 to select Read data 2 of the register file to an input of ALU (ALUSrc = 0 : ALU의 입력으로 레지스터 파일의 Read data 2를 선택한다.)
- MemtoReg = 0 to select the data value of ALU result to supply Write data of the register file (MemtoReg = 0: 레지스터 파일의 쓰기 데이터를 제공할 ALU 결과의 데이터 값을 선택한다.)
- For R-type instructions, sign extension unit and data memory are not used (R타입 명령어의 경우 부호 확장 단위와 데이터 메모리는 사용되지 않는다.)

1. 레지스터 파일 읽기 단계
    - $t2와 $t3의 값이 레지스터 파일에서 읽히는 단계이다.
    - 레지스터 파일은 $t2와 $t3의 값을 제공하기 위해 두 개의 읽기 포트를 가진다.

- 레지스터 파일 읽기 단계에서는 두 개의 읽기 포트가 활성화되고, $t2와 $t3의 값을 읽어온다.

2. Mux 선택 단계
    - 레지스터 파일에서 읽어온 $t2와 $t3의 값을 선택하는 Mux가 있다.
    - 이 Mux는 ALU에 입력될 값을 선택하기 위해 사용된다.
    - Mux의 입력은 레지스터 파일에서 읽어온 $t2와 $t3의 값이고, 제어 신호에 따라 두 입력 중 하나가 선택된다.
3. ALU 연산 단계
    - Mux를 통해 선택된 $t2와 $t3의 값이 ALU의 입력으로 전달된다.
    - ALU는 선택된 두 입력 값을 더하여 결과를 생성한다.
4. 레지스터 파일 쓰기 단계
    - ALU에서 생성된 결과가 $t1 레지스터에 쓰여진다.
    - 레지스터 파일은 쓰기 포트를 가지고 있으며, ALU의 결과를 $t1 레지스터에 쓰기 위해 사용된다.