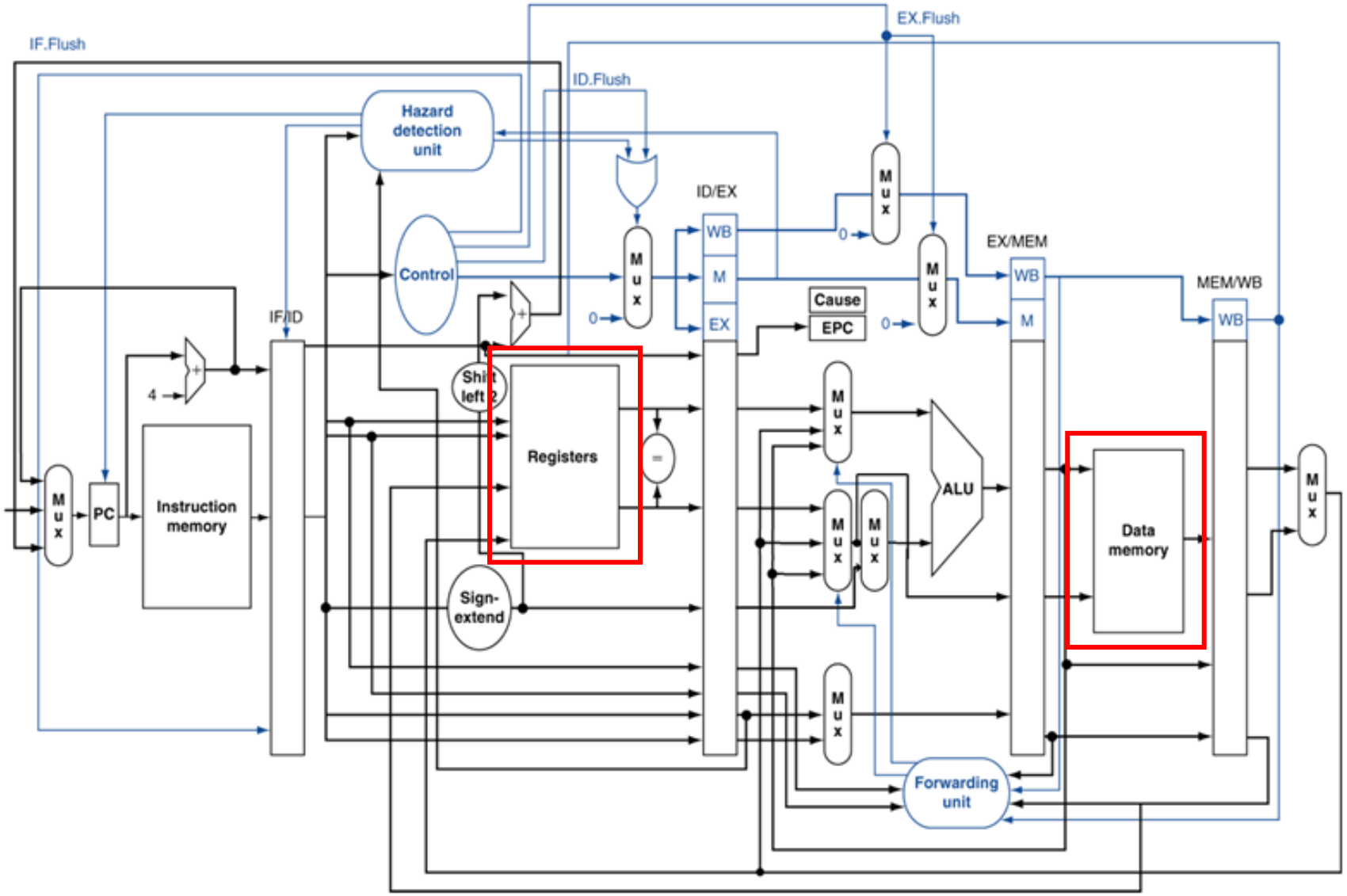
MIPS Register & Memory

조교 서유권

개요

- Introduction
- Register & Memory
- Load word & Store word
- MIPS MEM stage & WB stage
- 과제

Introduction



Register & Memory

- Register와 Memory 모두 **데이터를 저장**하는 장치
- Register
  - **작고 빠른** 저장 장치
  - MIPS는 **Register에 저장된 값을 가져와 연산** 수행
  - 주로 **32bit 레지스터 32개**로 구성 → 적을수록 빠르다
- Memory
  - **크고 느린** 저장 장치
  - Register보다 큰 데이터 저장

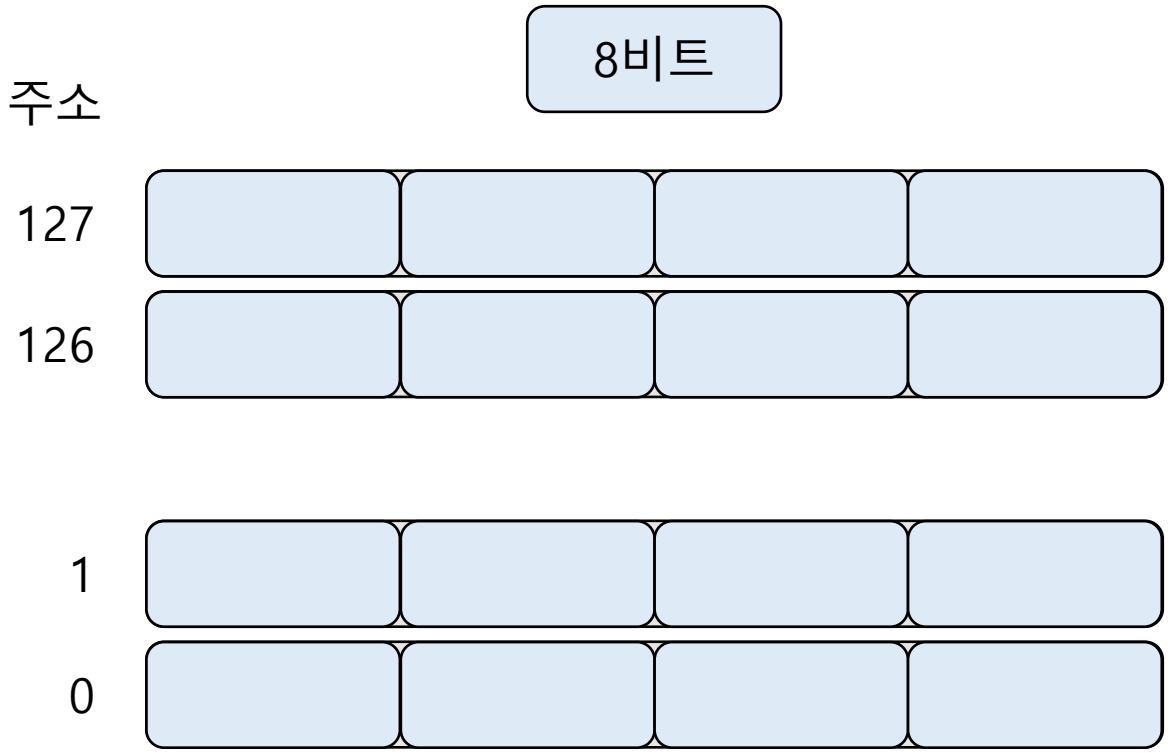
Register & Memory

- Register
  - **32bit Register X 32개**로 구현
  - 각 레지스터는 역할을 가지고 있음
    - $zero(0의 상수 저장)
    - $t(임시 변수 저장)
    - $s(임시 값 저장)

Name	Number	Use
$zero	$0	constant 0
$at	$1	assembler temporary
$v0–$v1	$2–$3	values for function returns and expression evaluation
$a0–$a3	$4–$7	function arguments
$t0–$t7	$8–$15	temporaries
$s0–$s7	$16–$23	saved temporaries
$t8–$t9	$24–$25	temporaries
$k0–$k1	$26–$27	reserved for OS kernel
$gp	$28	global pointer
$sp	$29	stack pointer
$fp	$30	frame pointer
$ra	$31	return address

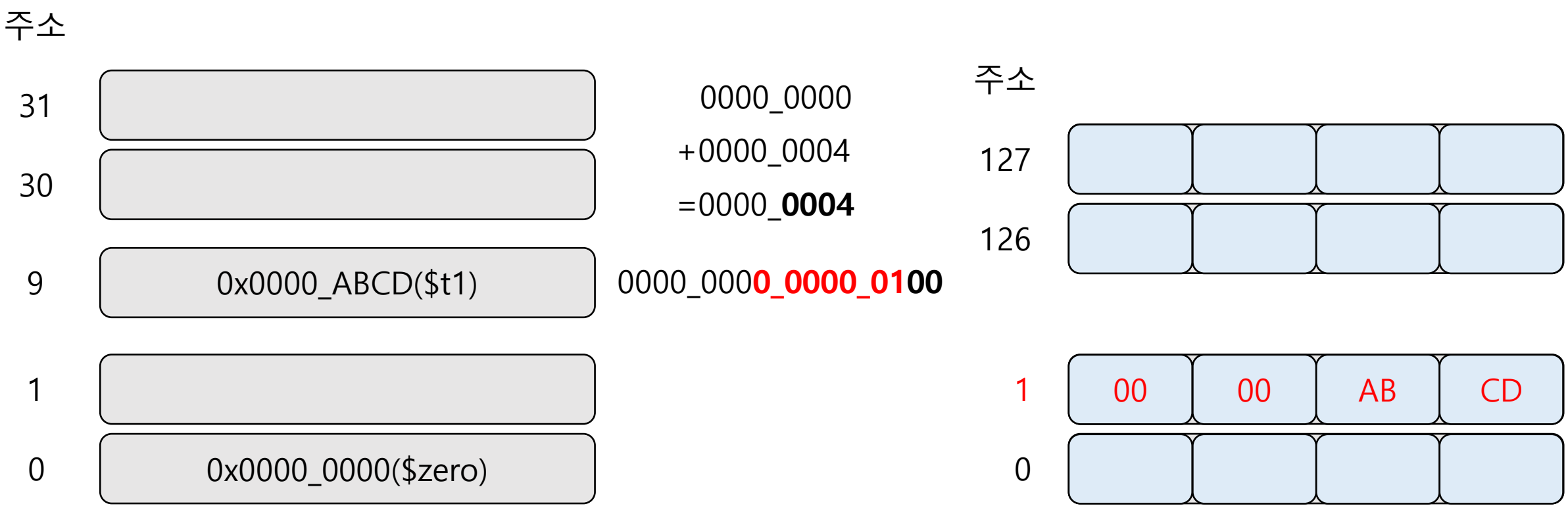
Register & Memory

- Memory
  - 각 주소에 8비트씩 데이터 저장
    - Offset 1 = 8비트
  - **128개의 주소 X 32비트 메모리** 구현



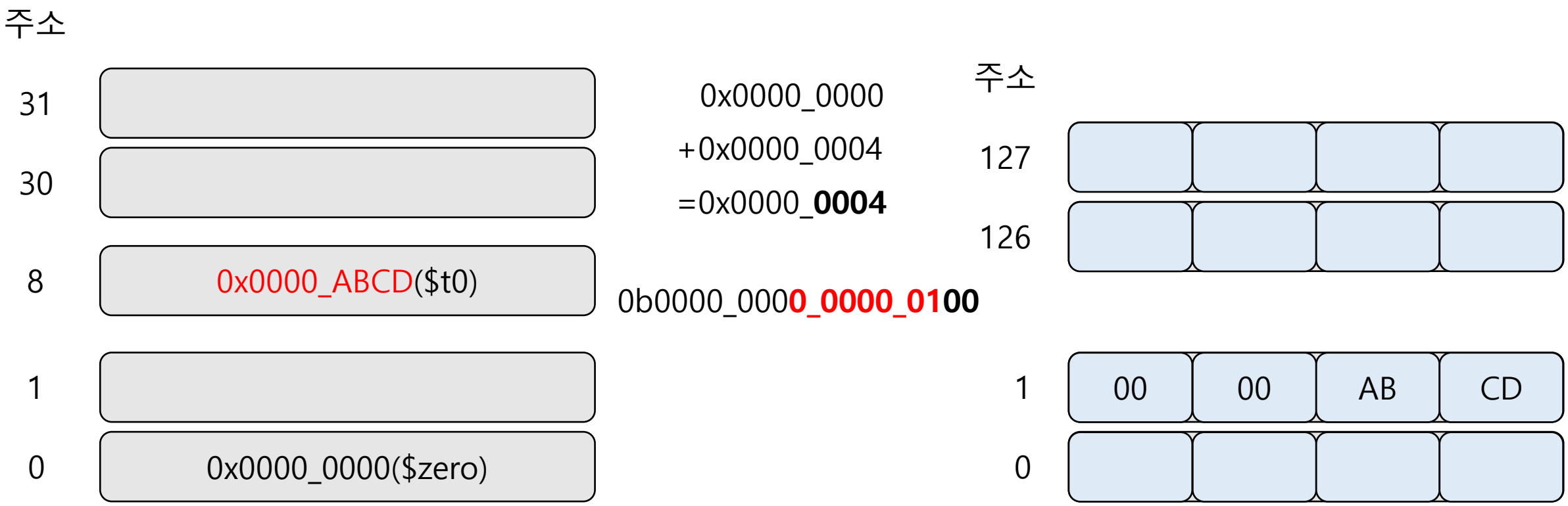
Load word & Store word

- sw $t1, 4($zero)
  - 메모리의 $zero + 4의 위치에 $t1에 저장된 값 저장 (레지스터 → 메모리)



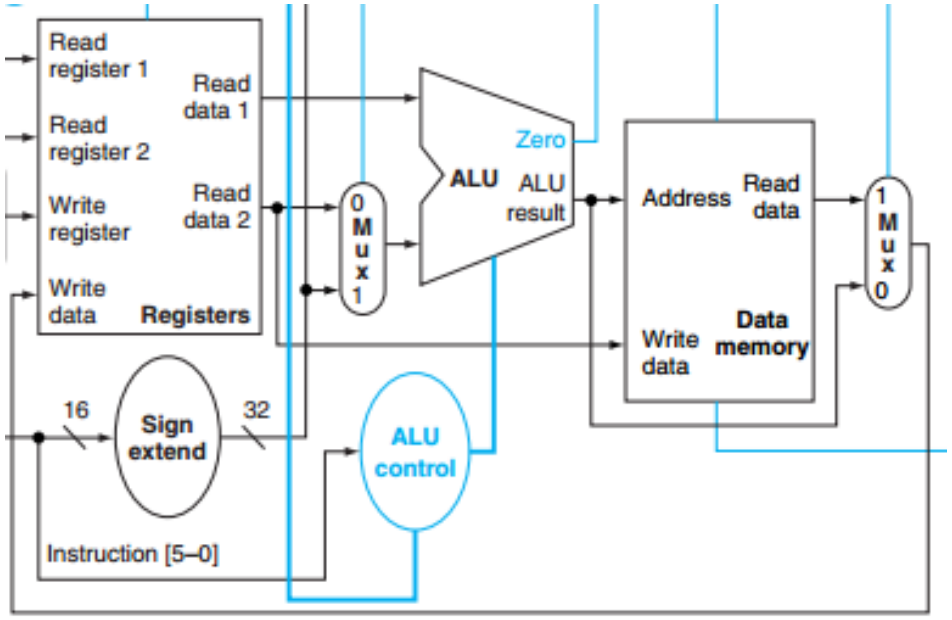
Load word & Store word

- lw $t0, 4($zero)
  - 메모리의 $zero + 4의 위치에 저장된 값을 레지스터 $t0에 저장 (메모리 → 레지스터)



MIPS MEM Stage & WB Stage

- MEM Stage (Memory Access)
  - Data Memory에 접근하여 값을 쓰거나 읽는 단계
- WB Stage (Write Back)
  - 연산의 결과를 레지스터에 쓰는 단계



과제

- Register 코드 작성

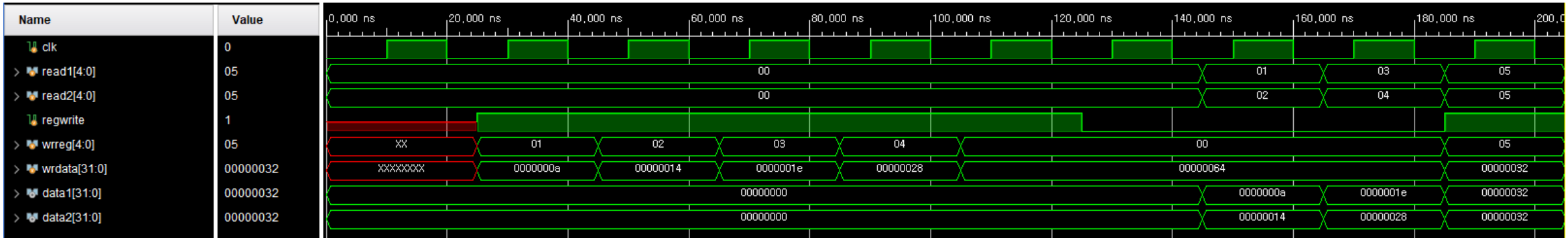
```
module regm(
    input wire              clk,
    input wire  [4:0]       read1, read2,   // READ ADDRESS
    output wire [31:0]      data1, data2,
    input wire              regwrite,
    input wire  [4:0]       wrreg,          // WRITE ADDRESS
    input wire  [31:0]      wrdata
    );

    // 32 bit register X 32
    reg [31:0]  mem [0:31];
```

```
module control(
                input  wire     [5:0]   opcode,
                output reg                      branch_eq, branch_ne,
                output reg [1:0]        aluop,
                output reg                      memread, memwrite, memtoreg,
                output reg                      regdst, regwrite, alusrc,
                output reg                      jump);
```

- Read 동작은 clk과 관계 없이 주소가 입력되면 바로 나오도록 설계
  - 0번째 Register Read 시, 항상 0 출력
  - Regwrite 신호가 1이고, read address가 wrreg일 때 대응되는 data에 wrdata 출력
- Write 동작은 posedge clk에 맞추어 값 할당(regwrite가 1일 때 동작)
  - 0번째 Register에는 값 쓰기 불가

과제

- Register Testbench (tb_regm.v)



과제

- Data Memory 코드 작성

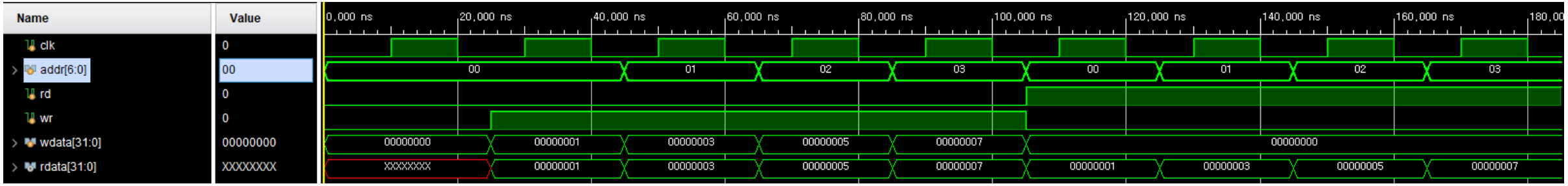
```
module dm(
    input wire              clk,
    input wire  [6:0]       addr,
    input wire              rd, wr,
    input wire  [31:0]      wdata,
    output wire [31:0]      rdata
    );

    // 32 bit memory with 128 address
    reg [31:0]  mem [0:127];
```

```
module control(
                input  wire     [5:0]   opcode,
                output reg                      branch_eq, branch_ne,
                output reg [1:0]        aluop,
                output reg                      memread, memwrite, memtoreg,
                output reg                      regdst, regwrite, alusrc,
                output reg                      jump);
```

- Read 동작은 clk과 관계 없이 주소가 입력되면 바로 나오도록 설계
  - Rd 신호가 1일 때, addr에서 읽기 동작
    - Rd 신호가 0일 때, wr 신호가 1이면 rdata에 wdata 출력 / rd와 wr 모두 0일 때는 자유롭게 설정
- Write 동작은 posedge clk에 맞추어 값 할당
  - Wr 신호가 1일 때 addr에 쓰기 동작

과제

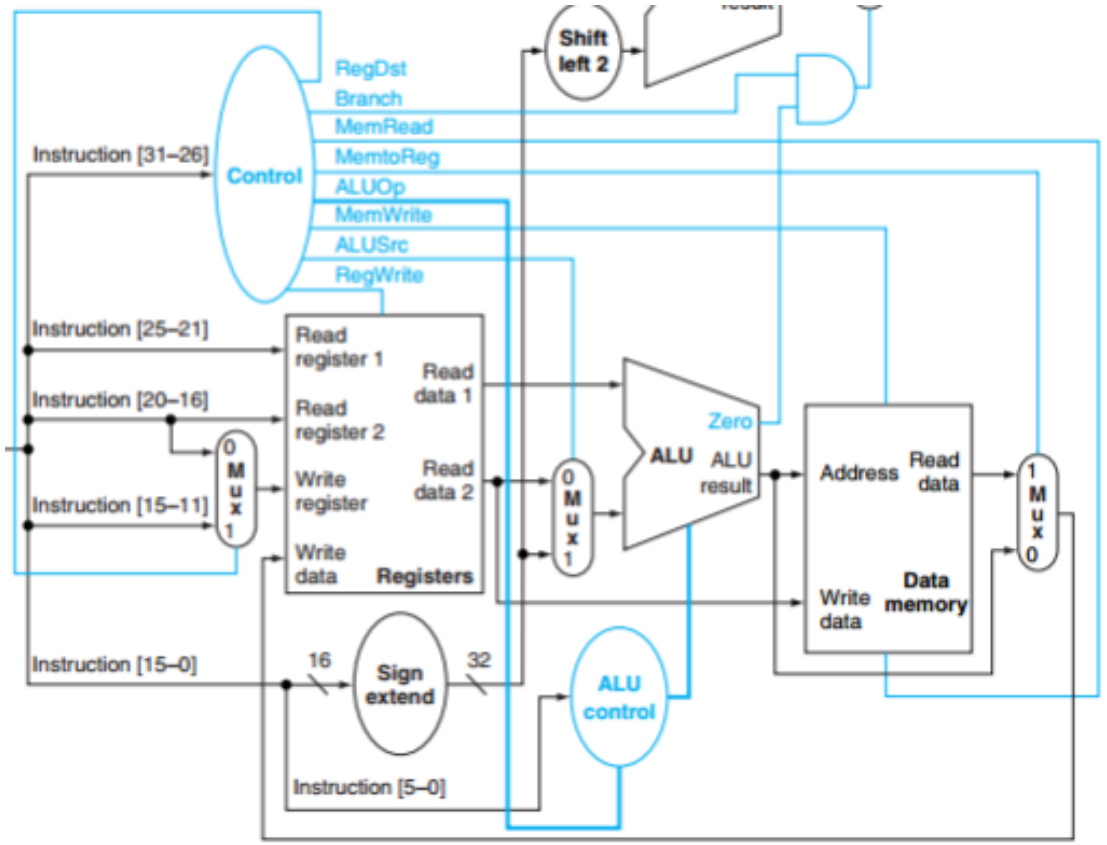
- Register Testbench (tb_dm.v)



과제(선택)

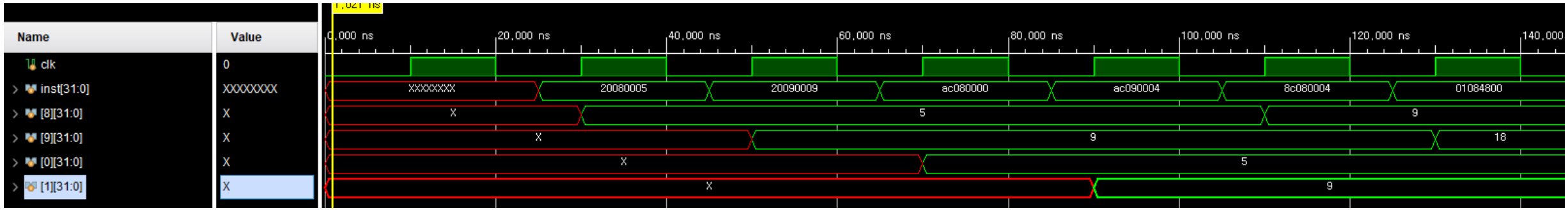
- Control + Register + ALU + Data Memory 설계
  - ID → EX → MEM → WB

```
module id_ex_mem_wb(
    input wire          clk,
    input wire  [31:0]  instruction
    );

    wire    [5:0]   opcode;
    wire    [4:0]   rs, rt;

    // R-Format
    wire    [4:0]   rd;
      wire    [4:0]   shamt;

    // I-Format
    wire    [15:0]  imm;
    wire    [31:0]  seimm;  // Sign-extend imm

    assign  opcode = instruction[31:26];
    assign  rs = instruction[25:21];
    assign  rt = instruction[20:16];
    assign  rd = instruction[15:11];
    assign  imm = instruction[15:0];
      assign  shamt = instruction[10:6];
    assign  seimm = {{16{instruction[15]}}, instruction[15:0]};
```



과제(선택)

- Register Testbench (tb_id_ex_mem_wb.v)




```
// addi $t0, $zero, 5
// addi $t1, $zero, 9
// sw $t0, 0($zero)
// sw $t1, 4($zero)
// lw $t0, 4($zero)
// add $t0, $t0, $t1
```