# 18011549 박태정

## Advanced ( 7점 )

## 3번 파일

```
if ( !(fp = fopen( filename: "3.txt",  mode: "r")) )
{
    printf("error: file open fail !!\n");
    exit(1);
}

while (feof(fp) == false)
```

```
Instruction = 08000010
>>Jump
PC : 64
Total cycle : 4

==================== REGISTER ====================
reg[00] = 00000000        reg[08] = 00003578        reg[16] = 00000040        reg[24] = 00000000
reg[01] = 00000000        reg[09] = 00041621        reg[17] = 00000000        reg[25] = 00000000
reg[02] = 00000000        reg[10] = 00083242        reg[18] = 00000000        reg[26] = 00000000
reg[03] = 00000000        reg[11] = 00000000        reg[19] = 00000000        reg[27] = 00000000
reg[04] = 00000000        reg[12] = 00000000        reg[20] = 00000000        reg[28] = 00000000
reg[05] = 00000000        reg[13] = 00000000        reg[21] = 00000000        reg[29] = 00000000
reg[06] = 00000000        reg[14] = 00000000        reg[22] = 00000000        reg[30] = 00000000
reg[07] = 00000000        reg[15] = 00000000        reg[23] = 00000000        reg[31] = 00000000
==================== REGISTER ====================

==================== MEMORY ====================
mem[00] = 000017387552        mem[32] = 000000000000
mem[04] = -01912078336        mem[36] = 000000000000
mem[08] = -01375010812        mem[40] = 000000003578
mem[12] = 000134217744        mem[44] = 000000000000
mem[16] = 000017395744        mem[48] = 000000000000
mem[20] = 000000000000        mem[52] = 000000000000
mem[24] = 000000000000        mem[56] = 000000000000
mem[28] = 000000000000        mem[60] = 000000000000
==================== MEMORY ====================
```

4번 입력 파일

```
if ( !(fp = fopen( filename: "4.txt",  mode: "r")) )
{
    printf("error: file open fail !!\n");
    exit(1);
}
```

```
Instruction = 08000010
>>Jump
PC : 64
Total cycle : 5

===================== REGISTER =====================
reg[00] = 00000000        reg[08] = 00003578        reg[16] = 00000040        reg[24] = 00000000
reg[01] = 00000000        reg[09] = 00041621        reg[17] = 00000000        reg[25] = 00000000
reg[02] = 00000000        reg[10] = 00000000        reg[18] = 00000000        reg[26] = 00000000
reg[03] = 00000000        reg[11] = 00000000        reg[19] = 00000000        reg[27] = 00000000
reg[04] = 00000000        reg[12] = 00041621        reg[20] = 00000000        reg[28] = 00000000
reg[05] = 00000000        reg[13] = 00000000        reg[21] = 00000000        reg[29] = 00000000
reg[06] = 00000000        reg[14] = 00041621        reg[22] = 00000000        reg[30] = 00000000
reg[07] = 00000000        reg[15] = 00000000        reg[23] = 00000000        reg[31] = 00000000
===================== REGISTER =====================

===================== MEMORY =====================
mem[00] = 000019554336        mem[32] = 000000000000
mem[04] = -01912078336        mem[36] = 000000000000
mem[08] = -01375010812        mem[40] = 000000003578
mem[12] = 000019755040        mem[44] = 000000000000
mem[16] = 000134217744        mem[48] = 000000000000
mem[20] = 000000000000        mem[52] = 000000000000
mem[24] = 000000000000        mem[56] = 000000000000
mem[28] = 000000000000        mem[60] = 000000000000
===================== MEMORY =====================
```

5번 입력 파일

```
if ( !(fp = fopen( filename: "5.txt",  mode: "r")) )
{
    printf("error: file open fail !!\n");
    exit(1);
}
```

```
Instruction = 08000010
>>Jump
PC : 64
Total cycle : 5

==================== REGISTER ====================
reg[00] = 00000000        reg[08] = 00041621        reg[16] = 00000040        reg[24] = 00000000
reg[01] = 00000000        reg[09] = 00041621        reg[17] = 00000000        reg[25] = 00000000
reg[02] = 00000000        reg[10] = 00000000        reg[18] = 00000000        reg[26] = 00000000
reg[03] = 00000000        reg[11] = 00000000        reg[19] = 00000000        reg[27] = 00000000
reg[04] = 00000000        reg[12] = 00041621        reg[20] = 00000000        reg[28] = 00000000
reg[05] = 00000000        reg[13] = 00000000        reg[21] = 00000000        reg[29] = 00000000
reg[06] = 00000000        reg[14] = 00041621        reg[22] = 00000000        reg[30] = 00000000
reg[07] = 00000000        reg[15] = 00000000        reg[23] = 00000000        reg[31] = 00000000
==================== REGISTER ====================

==================== MEMORY ====================
mem[00] = 000288030722        mem[32] = 000000000000
mem[04] = 000019554336        mem[36] = 000000000000
mem[08] = 000019755040        mem[40] = 000000003578
mem[12] = -01375141884        mem[44] = 000000041621
mem[16] = 000134217744        mem[48] = 000000000000
mem[20] = 000000000000        mem[52] = 000000000000
mem[24] = 000000000000        mem[56] = 000000000000
mem[28] = 000000000000        mem[60] = 000000000000
==================== MEMORY ====================
```

‘

1 번 입력 파일

```
if ( !(fp = fopen( filename: "1.txt",  mode: "r")) )
{
    printf("error: file open fail !!\n");
    exit(1);
}
```

```
Instruction = 08000010
>>Jump
PC : 64
Total cycle : 2

==================== REGISTER ====================
reg[00] = 00000000        reg[08] = 00041621        reg[16] = 00000040        reg[24] = 00000000
reg[01] = 00000000        reg[09] = 00041621        reg[17] = 00000000        reg[25] = 00000000
reg[02] = 00000000        reg[10] = 00000000        reg[18] = 00000000        reg[26] = 00000000
reg[03] = 00000000        reg[11] = 00000000        reg[19] = 00000000        reg[27] = 00000000
reg[04] = 00000000        reg[12] = 00000000        reg[20] = 00000000        reg[28] = 00000000
reg[05] = 00000000        reg[13] = 00000000        reg[21] = 00000000        reg[29] = 00000000
reg[06] = 00000000        reg[14] = 00000000        reg[22] = 00000000        reg[30] = 00000000
reg[07] = 00000000        reg[15] = 00000000        reg[23] = 00000000        reg[31] = 00000000
==================== REGISTER ====================

==================== MEMORY ====================
mem[00] = 000285802498        mem[32] = 000000000000
mem[04] = -01912078336        mem[36] = 000000000000
mem[08] = 000017387552        mem[40] = 000000003578
mem[12] = 000134217744        mem[44] = 000000000000
mem[16] = -01375076348        mem[48] = 000000000000
mem[20] = 000000000000        mem[52] = 000000000000
mem[24] = 000000000000        mem[56] = 000000000000
mem[28] = 000000000000        mem[60] = 000000000000
==================== MEMORY ====================
```

2번 입력 파일

```
if ( !(fp = fopen( filename: "2.txt",  mode: "r")) )
{
    printf("error: file open fail !!\n");
    exit(1);
}
```

```
Instruction = 08000010
>>Jump
PC : 64
Total cycle : 5

==================== REGISTER ====================
reg[00] = 00000000        reg[08] = 00003578        reg[16] = 00000040        reg[24] = 00000000
reg[01] = 00000000        reg[09] = 00041621        reg[17] = 00000000        reg[25] = 00000000
reg[02] = 00000000        reg[10] = 00045199        reg[18] = 00000000        reg[26] = 00000000
reg[03] = 00000000        reg[11] = 00000000        reg[19] = 00000000        reg[27] = 00000000
reg[04] = 00000000        reg[12] = 00000000        reg[20] = 00000000        reg[28] = 00000000
reg[05] = 00000000        reg[13] = 00000000        reg[21] = 00000000        reg[29] = 00000000
reg[06] = 00000000        reg[14] = 00000000        reg[22] = 00000000        reg[30] = 00000000
reg[07] = 00000000        reg[15] = 00000000        reg[23] = 00000000        reg[31] = 00000000
==================== REGISTER ====================

==================== MEMORY ====================
mem[00] = -01912078336        mem[32] = 000000000000
mem[04] = 000285802510        mem[36] = 000000000000
mem[08] = 000017387552        mem[40] = 000000003578
mem[12] = -01375076348        mem[44] = 000000045199
mem[16] = 000134217744        mem[48] = 000000000000
mem[20] = 000000000000        mem[52] = 000000000000
mem[24] = 000000000000        mem[56] = 000000000000
mem[28] = 000000000000        mem[60] = 000000000000
==================== MEMORY ====================
```

코드 전문

```c
#define _CRT_SECURE_NO_WARNINGS

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
/*************************************/

struct Control
{
    unsigned int RegDst;
    unsigned int Jump;
    unsigned int Branch;
    unsigned int MemRead;
    unsigned int MemtoReg;
    unsigned int ALUOp;
    unsigned int MemWrite;
    unsigned int ALUSrc;
    unsigned int RegWrite;
};

struct Reg_Read
{
    unsigned int Read_data_1;
    unsigned int Read_data_2;
};

struct ALU
{
    unsigned char zero; // 1: enable, 0: disable
    unsigned int ALU_result;
};

struct Control control;
struct Reg_Read reg_read;
struct ALU alu;
unsigned int mem[64] = { 0 };
unsigned int reg[32] = { 0 };

/*************************************/

unsigned int Inst_Fetch(unsigned int read_addr);
void Register_Read(unsigned int read_reg_1, unsigned int read_reg_2);
void Control_Signal(unsigned int opcode);
unsigned int ALU_Control_Signal(unsigned int signal);
void ALU_func(unsigned int ALU_control, unsigned int rd, unsigned int
plus);
unsigned int Memory_Access(unsigned int MemWrite, unsigned int MemRead,
unsigned int addr, unsigned int write_data);
void Register_Write(unsigned char RegWrite, unsigned int Write_reg,
unsigned int Write_data);
unsigned int Sign_Extend(unsigned int inst_16);
unsigned int Shift_Left_2(unsigned int inst);
unsigned int Add(unsigned int a, unsigned int b);
void print_reg_mem(void);

/*************************************/

int main(void)
{
    unsigned int pc = 0;
```

```
    FILE *fp;
    unsigned int inst = 0;
    unsigned int inst_31_26 = 0;
    unsigned int inst_25_21 = 0;
    unsigned int inst_20_16 = 0;
    unsigned int inst_15_11 = 0;
    unsigned int inst_15_0 = 0;
    unsigned int inst_ext_32 = 0;
    unsigned int inst_ext_shift = 0;
    unsigned int pc_add_4 = 0;
    unsigned int pc_add_inst = 0;
    unsigned int mux_result = 0;
    unsigned char ALU_control = 0;
    unsigned int inst_25_0 = 0;
    unsigned int inst_5_0 = 0;
    unsigned int jump_addr = 0;
    unsigned int mem_result = 0;
    int total_cycle = 0;

    // register initialization
    /*************************************/
    reg[8] = 41621;
    reg[9] = 41621;
    reg[16] = 40;
    /*************************************/

    // memory initialization
    /*************************************/
    mem[40] = 3578;

    if ( !(fp = fopen("2.txt", "r")) )
    {
        printf("error: file open fail !!\n");
        exit(1);
    }

    while (feof(fp) == false)
    {
        fscanf(fp, "%x", &inst);
        mem[pc] = inst;
        pc = pc + 4;
    }
    /*************************************/

    // control initialization
    /*************************************/
    control.RegDst = 0;
    control.Jump = 0;
    control.Branch = 0;
    control.MemRead = 0;
    control.ALUOp = 0;
    control.MemWrite = 0;
    control.ALUSrc = 0;
    control.RegWrite = 0;
    /*************************************/

    print_reg_mem();

    printf("\n ***** Processor START !!! ***** \n");

    pc = 0;
```

```
    while (pc < 64){

        // instructio fetch
        inst = Inst_Fetch(pc);
        printf("Instruction = %08x \n", inst);

        // pc +4
        pc = Add(pc, 4);

        // instruction decode
        inst_31_26 = inst >> 26; ////Operation Code -> MIPS 상 어떤
명령어인지 특정할 수 있게 함.
        inst_25_21 = (inst & 0x03e00000) >> 21; ////rs 첫번째 인자
        inst_20_16 = (inst & 0x001f0000) >> 16; //// rt 두번째 인자
        inst_15_11 = (inst & 0x0000f800) >> 11; //// rd destination
        inst_15_0 = inst & 0x0000ffff; //// 조건분기문에 사용 함.
        inst_25_0 = inst & 0x03ffffff; /////분기문일때 사용하는 주
        inst_5_0  = inst & 0x0000003f; ///// func

        //printf("\n%x, %x, %x, %x, %x, %x, %x\n\n", inst_31_26,
inst_25_21, inst_20_16, inst_15_11, inst_5_0, inst_15_0, inst_25_0);

        //// register read
        //// create control signal
        //// create ALU control signal
        //// ALU

        if(inst_31_26==0) { //// 산술연산
            Register_Read(inst_25_21, inst_20_16);
            ////reg_read 구조체 사용
            Control_Signal(0);
            //// register write
            ALU_func(ALU_Control_Signal(inst_5_0),inst_15_11,0);

        }

        //// load word , store word
        if(inst_31_26==35 || inst_31_26==43) {
            Register_Read(inst_25_21, inst_20_16);
            if(inst_31_26==35) { ////load word
                Control_Signal(35);
                Register_Read(inst_25_21,inst_20_16);


ALU_func(ALU_Control_Signal(35),inst_20_16,Sign_Extend(inst_15_0)+reg[inst_
25_21]);
                printf(">>Load Word\n");
            }
            else { //// store word
                Control_Signal(43);
                Register_Read(inst_25_21,inst_20_16);


ALU_func(ALU_Control_Signal(43),reg[inst_25_21]+Sign_Extend(inst_15_0),inst
_20_16);
                printf(">>Store Word\n");
```

```
            }
            //// lw 와 rw 위치 헷갈리던
        }
        if (inst_31_26 == 4) { ///Branch Equal
            Control_Signal(4);
            Register_Read(inst_25_21, inst_20_16);
            ALU_func(ALU_Control_Signal(4),pc,0);
            if(reg_read.Read_data_1 == 0){
                pc = (pc) + Shift_Left_2(Sign_Extend(inst_15_0));

            }
        }

        if (inst_31_26 == 2) { //// Jump
           Control_Signal(2);
            pc=(pc+4)&0xf0000000 | ( inst_25_0 << 2 );
           //pc=pc+Shift_Left_2(Sign_Extend(inst_25_0));
           printf(">>Jump\n");
        }
        // memory access
        total_cycle++;
        // result
        /*********************************/
        printf("PC : %d \n", pc);
        printf("Total cycle : %d \n", total_cycle);
        print_reg_mem();
        /*********************************/
    }
    printf("\n ***** Processor END !!! ***** \n");
    return 0;
}

unsigned int Inst_Fetch(unsigned int read_addr)
{
    return mem[read_addr];
}

void Register_Read(unsigned int read_reg_1, unsigned int read_reg_2)
{
   reg_read.Read_data_1=reg[read_reg_1];
   reg_read.Read_data_2=reg[read_reg_2];
}

void Control_Signal(unsigned int opcode)
{
    if(opcode==0) { //// 산술연
        control.ALUOp = 2;
        ////Opcode 에 의해 Control 필드가 결정된다.
        control.RegDst = 0;
        control.ALUSrc = 0;
        control.MemtoReg = 0;
        control.RegWrite = 1;
        control.MemRead = 0;
        control.MemWrite = 0;
        control.Branch = 0;
        control.Jump = 0;
    }

    if (opcode == 35) { //// lw ( Load Word )
        control.ALUOp = 0;
```

```
        control.RegDst = 0;
        control.ALUSrc = 1;
        control.MemtoReg = 1;
        control.RegWrite = 1;
        control.MemRead = 1;
        control.MemWrite = 0;
        control.Branch = 0;
        control.Jump=0;
    }
    if (opcode == 43) { //// SW ( Store Word )
        control.ALUOp = 0;
        control.RegDst = 'x';
        control.ALUSrc = 1;
        control.MemtoReg = 'x';
        control.RegWrite = 0;
        control.MemRead = 0;
        control.MemWrite = 1;
        control.Branch = 0;
        control.Jump=0;
    }

    if (opcode == 4) { ///Branch Equal
        control.ALUOp = 1;
        control.RegDst = 'x';
        control.ALUSrc = 0;
        control.MemtoReg = 'x';
        control.RegWrite = 0;
        control.MemRead = 0;
        control.MemWrite = 0;
        control.Branch = 1;
        control.Jump=0;
    }
    if (opcode== 2) { //// Jump
        control.ALUOp = 2;
        control.RegDst = 'x';
        control.ALUSrc = 'x';
        control.MemtoReg = 'x';
        control.Jump = 1;
    }
}

unsigned int ALU_Control_Signal(unsigned int signal)
{
    if(signal == 32) { //// add
        printf(">>ADD\n");
        return 2;
    }
    if(signal == 34){ ////sub
        printf(">>SUB\n");
        return 6;
    }
    if(signal == 40){ //// AND
        printf(">>AND\n");
         return 0;
    }
    if(signal == 41){ ////OR
        printf(">>OR\n");
        return 1;
    }
    if(signal == 46) { //// set on less than
        printf(">>SLT\n");
```

```
        return 7;
    }
    if(signal==35) { //// Load Word or
        printf(">>LW\n");
        return 101;
    }
    if(signal==43) { //// Save Word
        printf(">>SW\n");
        return 102;
    }
    if(signal==4) { ///branch
        printf(">>BEQ\n");
        return 103;
    }
}

void ALU_func(unsigned int ALU_control, unsigned int rd,unsigned int plus)
{
    unsigned int result;
  ////add
    if(ALU_control==2) {
        result = reg_read.Read_data_1 + reg_read.Read_data_2;
        alu.ALU_result=result;
        Register_Write(control.RegWrite,rd,alu.ALU_result);
    }
  ////sub
     if(ALU_control==6){
        result = reg_read.Read_data_1 - reg_read.Read_data_2;
        alu.ALU_result=result;
        Register_Write(control.RegWrite,rd,alu.ALU_result);
    }
  ///AND
    if(ALU_control==0){
        result = reg_read.Read_data_1 & reg_read.Read_data_2;
        alu.ALU_result=result;
        Register_Write(control.RegWrite,rd,alu.ALU_result);
    }
  ///OR
    if(ALU_control==1){
        result = reg_read.Read_data_1 | reg_read.Read_data_2;
        alu.ALU_result=result;
        Register_Write(control.RegWrite,rd,alu.ALU_result);
    }
  ///slt , set less than
    if(ALU_control==46){
        if(reg_read.Read_data_1 < reg_read.Read_data_2)
            result=1;
        else
            result=0;
        alu.ALU_result=result;
        Register_Write(control.RegWrite,rd,alu.ALU_result);
    }

    ///Load Word
    if(ALU_control==101){

//Register_Write(control.RegWrite,rd,reg_read.Read_data_1+reg_read.Read_dat
a_2);
        Memory_Access(control.MemWrite,control.MemRead,rd,mem[plus]);
    }
    ////Store Word
```

```
    if(ALU_control==102){
        Memory_Access(control.MemWrite,control.MemRead,rd,reg[plus]);
    }

////////////////////////////////////////////////////////////////////////////////
//////////////
    ////        Store Word 에서 , 어떻게 offset 이 0 인데 mem[40] 의 정보를
가져올 수 있는지 ?       ////
    ////        주소 지정 및 데이터 입력 위치를 어떻게 해야하는지 고민 중이었음.
////

////////////////////////////////////////////////////////////////////////////////
//////////////

    ////branch
    if(ALU_control==103){
        //rd = pc
        if(reg_read.Read_data_1-reg_read.Read_data_2==0){
            reg_read.Read_data_1=0;
        }
    }
}

unsigned int Memory_Access(unsigned int MemWrite, unsigned int MemRead,
unsigned int addr, unsigned int write_data)
{
   if(MemRead==1 && MemWrite==0){ ////Load Word
       reg[addr]=write_data;
   }
   if(MemRead==0 && MemWrite==1){ ////Store Word
       mem[addr]=write_data;
   }
}

void Register_Write(unsigned char RegWrite, unsigned int Write_reg,
unsigned int Write_data)
{
   if(RegWrite == 1)
       reg[Write_reg]=Write_data;
}

unsigned int Sign_Extend(unsigned int inst_16)
{
    unsigned int inst_32 = 0;
    if ((inst_16 & 0x00008000)) // minus
    {
        inst_32 = inst_16 | 0xffff0000;
    }
    else // plus
    {
        inst_32 = inst_16;
    }

    return inst_32;
}

unsigned int Shift_Left_2(unsigned int inst)
{
    return inst << 2;
}
```

```
unsigned int Add(unsigned int a, unsigned int b)
{
    return a+b;
}

void print_reg_mem(void)
{
    int reg_index = 0;
    int mem_index = 0;

    printf("\n==================== REGISTER ====================\n");

    for (reg_index = 0; reg_index < 8; reg_index++)
    {
        printf("reg[%02d] = %08d        reg[%02d] = %08d        reg[%02d] =
%08d        reg[%02d] = %08d \n",
               reg_index, reg[reg_index], reg_index+8, reg[reg_index+8],
reg_index+16, reg[reg_index+16], reg_index+24, reg[reg_index+24] );
    }

    printf("==================== REGISTER ====================\n");

    printf("\n==================== MEMORY ====================\n");

    for (mem_index = 0; mem_index < 32; mem_index = mem_index + 4)
    {
        printf("mem[%02d] = %012d        mem[%02d] = %012d \n",
               mem_index, mem[mem_index], mem_index + 32, mem[mem_index +
32]);
    }
    printf("==================== MEMORY ====================\n");
}
```

# 느낀점.

시간 상 challenge 를 도전하지 못한것에 아쉬움을 느낍니다. 어렵긴 하지만 확실히 실습이 이론을 익히는데 큰 도움을 주는 것 같습니다. 이런 실습은 앞으로도 컴퓨터 구조를 수강하는 학우들에게 큰 도움이 될 것 같다는 생각을 하게 되는 새벽입니다.