컴퓨터 구조론

- lab1 -

2024 . 04 . 14 제출

담당교수	남재현 교수님
학 과	컴퓨터공학과
학 번	32234743
이 름	최현우

1. Introduction(과제 개요)

이 과제에서는 MIPS 어셈블리 언어의 기본적인 명령어를 해석하고 실행하는 시뮬레이터를 C 언어를 사용하여 구현하였습니다. 파일에서 MIPS 명령어를 읽어 해당 명령어를 수행하고, 결과를 출력합니다.

2. Background(배경 지식)

MIPS (Microprocessor without Interlocked Pipelined Stages)는 RISC (Reduced Instruction Set Computer) 아키텍처를 기반으로 한 프로세서 아키텍처입니다. 이 시뮬레이터는 MIPS의 주요 명령어들을 구현하여, 기본적인 연산 및 제어를 시뮬레이션 합니다.

LW (Load Word): 주어진 값을 특정 레지스터에 로드합니다.
ADD, SUB, MUL, DIV: 두 레지스터의 값을 가지고 산술 연산을 수행하고, 결과를 다른 레지스터에 저장합니다.

- ADD: 두 레지스터의 합
- SUB: 두 레지스터의 차
- MUL: 두 레지스터의 곱
- DIV: : 두 레지스터의 몫

SLT (Set on Less Than): 두 레지스터의 값을 비교하고, 하나가 다른 하나보다 작을 경우 1을, 그렇지 않을 경우 0을 저장합니다.

JMP: 특정 라인으로 점프합니다.

BEQ, BNE: 조건에 따라 또는 무조건적으로 특정 라인으로 점프합니다.

- BEQ: 같은 경우 해당조건 수행
- BNE: 다른 경우 해당조건 수행

프로그램 구현을 위해 위와 같은 기본 로직에 대한 이해도를 필요로 합니다.

3. Design(디자인)

주요 명령어 처리 로직은 다음과 같습니다:
파일 입력으로 받아 한 라인씩 명령어를 확인합니다.

1) 명령어가 LW인 경우:
opcode | destination | value

2) 명령어가 JMP인 경우:
opcode | value(jump line)

3) 명령어가 BNE, BEQ인 경우:
opcode | value1 | value2 | value3(jump line)

4) 명령어가 ADD, SUB, MUL, DIV, SLT인 경우:
opcode | destination | value1 | value2

5) 명령어가 NOP인 경우
opcode

먼저 명령어 형식에 따라 구성하는 opcode와 operand의 개수가 달라질 수 있습니다. 또한 특정 opcode에 따른 operand가 각 의미하는 값과 배치가 달라질 수 있어 위와 같은 경우로 나누어 각 연산을 처리하는 코드를 작성하였습니다.

4. 구현

4.1. 구현 내용

- 파일 입력

메인 함수의 인자로 파일이름을 입력받아 열고, 예외처리 후 각 명령어 처리를 위해 파일의 라인수 파악

```
int main(int argc, char *argv[]) {
    if (argc < 2) {
        fprintf(stderr, "Usage: %s <filename>\n", argv[0]);
        return 1;
    }

    FILE *file = fopen(argv[1], "r");
    if (file == NULL) {
        perror("파일을 열수 없음");
        return 1;
    }

    char lines[MAX_LINES][100]; // jmp, bne, beq를 위한 라인수 저장배열
    int lineCount = 0;
    while (fgets(lines[lineCount], sizeof(lines[lineCount]), file)) {
        lineCount++;
    }
    fclose(file);
```

- 명령어가 LW인 경우:

현재 라인의 개수를 확인하고 명령어가 "LW"인 경우 해당 s 레지스터에 읽어드린 value값 저장 후 다음 줄 실행

```
// 명령어가 LW인 경우
if (sscanf(line, "%s %s %x", op, dst, &value) == 3 && strcmp(op, "LW") == 0) {
    int dstIndex = dst[1] - '0';
    if (dst[0] == 's' && dstIndex >= 0 && dstIndex < MAX_S_REGISTERS) { //해당 s레지스터 위치에 값 저장
        s_registers[dstIndex] = value;
        printf("Loaded %d to s%d\n", value, dstIndex);
    }
    continue;
}
```

- 명령어가 JMP인 경우:
현재 라인의 개수를 확인하고 명령어가 "JMP"인 경우 읽어드린 점프라인 줄수를 0기반의 인덱스로 변환하여 해당 줄수의 – 1값의 위치로 변경하여 현재 라인수로 저장 -> continue를 실행하면서 for문의 저장된 현재라인 + 1값으로 이동되어 읽어드린 점프 라인수로 이동

```
// 명령어가 JMP인 경우
if (sscanf(line, "%s %x", op, &jumpLine) == 2 && strcmp(op, "JMP") == 0) {
    if (jumpLine > 0 && jumpLine <= lineCount) {
        currentLine = jumpLine - 2;
        // 읽어드린 줄번호의 0기반 인덱스 이전번호로 이동(continue시 currentLine++을 수행하기 때문)
    } else {
        printf("Invalid jump line %d\n", jumpLine);
        break; // 점프 범위가 벗어나면 프로그램 종료
    }
    continue;
}
```

- 명령어가 BNE, BEQ인 경우:
조건을 비교하기 위한 두 값을 index1, index2에 불러와 비교 수행
BEQ의 경우 값이 동일한 경우 읽어드린 라인으로 이동
BNE의 경우 값이 동일하지 않은 경우 읽어드린 라인으로 이동

```
// 명령어가 BNE, BEQ인 경우
if (sscanf(line, "%s %s %s %x", op, src1, src2, &jumpLine) == 4 && (strcmp(op, "BEQ") == 0 || strcmp(op, "BNE") == 0)) {
    index1 = src1[0] == 's' ? s_registers[src1[1] - '0'] : (src1[0] == 't' ? t_registers[src1[1] - '0'] : 0);
    index2 = src2[0] == 's' ? s_registers[src2[1] - '0'] : (src2[0] == 't' ? t_registers[src2[1] - '0'] : 0);
    // 읽어드린 레지스터가 s인 경우, t 인경우, zero인 경우 저장했던 해당 위치의 값을 index1과 index2에 받아옴

    if (strcmp(op, "BEQ") == 0 && index1 == index2) {   //BEQ의 경우 -> 값이 동일한 경우
        currentLine = jumpLine - 2; // 읽어드린 줄번호의 0기반 인덱스 이전번호로 이동
        printf("%s: %s(%d) == %s(%d) and Jumped line %d\n", op, src1, index1, src2, index2, jumpLine);
        continue;
    } else if (strcmp(op, "BNE") == 0 && index1 != index2) {    //BNE의 경우 -> 값이 동일하지 않은 경우
        currentLine = jumpLine - 2; //읽어드린 줄번호의 0기반 인덱스 이전번호로 이동
        printf("%s: %s(%d) != %s(%d) and Jumped line %d\n", op, src1, index1, src2, index2, jumpLine);
        continue;
    }
    printf("%s: %s(%d) and %s(%d)\n", op, src1, index1, src2, index2);
```

- 명령어가 ADD, SUB, MUL, DIV, SLT인 경우:
연산후 저장할 목적지 주소를 dest 값에 저장함
연산을 위한 두 값을 index1, index2에 불러옴
각 ADD, SUB, MUL, DIV, SLT 연산 수행하여 해당 결과 값 저장

```
// BNE, BEQ 이외의 명령어인 경우 ADD, SUB, MUL, DIV, SLT 명령어 처리
} else if (sscanf(line, "%s %s %s %s", op, dst, src1, src2) == 4) {
    int *dest = (dst[0] == 't') ? &t_registers[dst[1] - '0'] : (dst[0] == 's' ? &s_registers[dst[1] - '0'] : &r0);
    // 계산 처리후 저장할 레지스터 t인경우, s인경우, r0인 경우
    index1 = src1[0] == 's' ? s_registers[src1[1] - '0'] : (src1[0] == 't' ? t_registers[src1[1] - '0'] : 0);
    index2 = src2[0] == 's' ? s_registers[src2[1] - '0'] : (src2[0] == 't' ? t_registers[src2[1] - '0'] : 0);
    // 읽어드린 레지스터가 s인 경우, t 인경우, zero인 경우 저장했던 해당 위치의 값을 index1과 index2에 받아옴

    if (strcmp(op, "ADD") == 0) {           // ADD의 경우 덧셈
        *dest = index1 + index2;
    } else if (strcmp(op, "SUB") == 0) {    // SUB의 경우 뺄셈
        *dest = index1 - index2;
    } else if (strcmp(op, "MUL") == 0) {    // MUL의 경우 곱셈
        *dest = index1 * index2;
    } else if (strcmp(op, "DIV") == 0) {    // DIV의 경우 나눗셈
        if (index2 != 0) {
            *dest = index1 / index2;
        } else {    //분모가 0이 아닌 경우 예외 처리
            printf("Cannot divide by zero.\n");
            continue;
        }
    } else if (strcmp(op, "SLT") == 0) {    // SLT의 경우, index1 <index2 이면 dst에 1, 아니면 0 저장
        *dest = (index1 < index2) ? 1 : 0;
    }
    printf("%sed %s(%d) and %s(%d) and changed %s to %d\n", op, src1, index1, src2, index2, dst, *dest);
```

- 명령어가 NOP인 경우

```
} else if (strcmp(line, "NOP") == 0) {      // NOP의 경우
    printf("No operation\n");
}
```

- 최종결과 값
r0에 저장된 값 출력 후 프로그램 종료

```
    // 최종 결과 값 출력
    printf("Final result: r0 = %d\n", r0);
    return 0;
}
```

4.2. (O, X) Table

기능	구현여부
LW	O
ADD	O
SUB	O
MUL	O
DIV	O
JMP	O
BEQ	O
BNE	O
SLT	O

4.3. 구현 결과(테스트 케이스 및 결과 화면)

- 테스트케이스1 (테스트 기능: LW, ADD)

test.txt

```
1    LW s0 0xF
2    LW s1 0x4
3    ADD r0 s0 s1
```

- 결과화면

```
koicloud@ca-32234743:~/vscode$ ./32234743 test.txt
Loaded 15 to s0
Loaded 4 to s1
ADDed s0(15) and s1(4) and changed r0 to 19
Final result: r0 = 19
```

- 테스트케이스2 (테스트 기능: SUB)

test.txt

```
1    LW s0 0x1
2    LW s1 0x2
3    LW s2 0x3
4    LW s3 0x4
5    ADD t0 s0 s1
6    ADD t1 s2 s3
7    SUB r0 t0 t1
```

- 결과화면

```
koicloud@ca-32234743:~/vscode$ ./32234743 test.txt
Loaded 1 to s0
Loaded 2 to s1
Loaded 3 to s2
Loaded 4 to s3
ADDed s0(1) and s1(2) and changed t0 to 3
ADDed s2(3) and s3(4) and changed t1 to 7
SUBed t0(3) and t1(7) and changed r0 to -4
Final result: r0 = -4
```

- 테스트케이스3 (테스트 기능: MUL)

test.txt

```
1    LW s0 0x2
2    LW s1 0x4
3    MUL r0 s0 s1
4
```

- 결과화면

```
koicloud@ca-32234743:~/vscode$ ./32234743 test.txt
Loaded 2 to s0
Loaded 4 to s1
MULed s0(2) and s1(4) and changed r0 to 8
Final result: r0 = 8
```

- 테스트케이스4 (테스트 기능: DIV)

test.txt
```
1   LW s0 0x2
2   LW s1 0x4
3   DIV r0 s0 s1
4
```

- 결과화면

```
koicloud@ca-32234743:~/vscode$ ./32234743 test.txt
Loaded 2 to s0
Loaded 4 to s1
DIVed s0(2) and s1(4) and changed r0 to 0
Final result: r0 = 0
```

- 테스트케이스5 (테스트 기능: DIV)

test.txt
```
1   LW s0 0x2
2   LW s1 0x4
3   DIV r0 s1 s0
4
```

- 결과화면

```
koicloud@ca-32234743:~/vscode$ ./32234743 test.txt
Loaded 2 to s0
Loaded 4 to s1
DIVed s1(4) and s0(2) and changed r0 to 2
Final result: r0 = 2
```

- 테스트케이스6 (테스트 기능: JMP)

test.txt
```
1   LW s0 0x5
2   LW s1 0xA
3   ADD t0 s0 s1
4   JMP 0x8
5   LW s0 0x1
6   LW s1 0x2
7   ADD t0 s0 s1
8   LW s2 0x3
9   MUL r0 t0 s2
10
```

- 결과화면

```
koicloud@ca-32234743:~/vscode$ ./32234743 test.txt
Loaded 5 to s0
Loaded 10 to s1
ADDed s0(5) and s1(10) and changed t0 to 15
Loaded 3 to s2
MULed t0(15) and s2(3) and changed r0 to 45
Final result: r0 = 45
```

- 테스트케이스7 (테스트 기능: BEQ - 같지 않은 경우)

test.txt

```
1   LW s0 0x5
2   LW s1 0xA
3   BEQ s0 s1 0x6
4   ADD r0 s0 zero
5   JMP 0x7
6   ADD r0 s1 zero
7   NOP
```

- 결과화면

```
koicloud@ca-32234743:~/vscode$ ./32234743 test.txt
Loaded 5 to s0
Loaded 10 to s1
BEQ: s0(5) and s1(10)
ADDed s0(5) and zero(0) and changed r0 to 5
No operation
Final result: r0 = 5
```

- 테스트케이스8 (테스트 기능: BEQ – 같은 경우)

test.txt

```
1   LW s0 0x5
2   LW s1 0x5
3   LW s2 0x7
4   BEQ s0 s1 0x6
5   ADD r0 s0 zero
6   JMP 0x7
7   ADD r0 s1 s2
8   NOP
```

- 결과화면

```
koicloud@ca-32234743:~/vscode$ ./32234743 test.txt
Loaded 5 to s0
Loaded 5 to s1
Loaded 7 to s2
BEQ: s0(5) == s1(5) and Jumped line 6
ADDed s1(5) and s2(7) and changed r0 to 12
No operation
Final result: r0 = 12
```

- 테스트케이스9 (테스트 기능: BNE – 같지 않은 경우)

test.txt

```
1    LW s0 0x5
2    LW s1 0xA
3    BNE s0 s1 0x6
4    ADD r0 s0 zero
5    JMP 0x7
6    ADD r0 s1 zero
7    NOP
```

- 결과화면

```
koicloud@ca-32234743:~/vscode$ ./32234743 test.txt
Loaded 5 to s0
Loaded 10 to s1
BNE: s0(5) != s1(10) and Jumped line 6
ADDed s1(10) and zero(0) and changed r0 to 10
No operation
Final result: r0 = 10
```

- 테스트케이스10 (테스트 기능: BNE – 같지 않은 경우)

test.txt

```
1    LW s0 0x5
2    LW s1 0xA
3    LW s2 0x1
4    BNE s0 s1 0x6
5    ADD r0 s0 zero
6    JMP 0x7
7    ADD r0 s1 s2
8    NOP
```

- 결과화면

```
koicloud@ca-32234743:~/vscode$ ./32234743 test.txt
Loaded 5 to s0
Loaded 10 to s1
Loaded 1 to s2
BNE: s0(5) != s1(10) and Jumped line 6
ADDed s1(10) and s2(1) and changed r0 to 11
No operation
Final result: r0 = 11
```

- 테스트케이스11 (테스트 기능: BNE – 같은 경우)

test.txt

```
1   LW s0 0x5
2   LW s1 0x5
3   LW s2 0x1
4   BNE s0 s1 0x6
5   ADD r0 s0 zero
6   JMP 0x8
7   ADD r0 s1 s2
8   NOP
```

- 결과화면

```
koicloud@ca-32234743:~/vscode$ ./32234743 test.txt
Loaded 5 to s0
Loaded 5 to s1
Loaded 1 to s2
BNE: s0(5) and s1(5)
ADDed s0(5) and zero(0) and changed r0 to 5
No operation
Final result: r0 = 5
```

- 테스트케이스12 (테스트 기능: SLT – a > b인 경우)

test.txt

```
1   LW s0 0x5
2   LW s1 0xA
3   SLT t0 s1 s0
4   BNE t0 zero 0x7
5   ADD r0 s0 zero
6   JMP 0x8
7   ADD r0 s1 zero
8   NOP
```

- 결과화면

```
koicloud@ca-32234743:~/vscode$ ./32234743 test.txt
Loaded 5 to s0
Loaded 10 to s1
SLTed s1(10) and s0(5) and changed t0 to 0
BNE: t0(0) and zero(0)
ADDed s0(5) and zero(0) and changed r0 to 5
Final result: r0 = 5
```

- 테스트케이스12 (테스트 기능: SLT – a < b인 경우)

test.txt

```
1   LW s0 0x5
2   LW s1 0xA
3   SLT t0 s0 s1
4   BNE t0 zero 0x7
5   ADD r0 s0 zero
6   JMP 0x8
7   ADD r0 s1 zero
8   NOP
```

- 결과화면

```
koicloud@ca-32234743:~/vscode$ ./32234743 test.txt
Loaded 5 to s0
Loaded 10 to s1
SLTed s0(5) and s1(10) and changed t0 to 1
BNE: t0(1) != zero(0) and Jumped line 7
ADDed s1(10) and zero(0) and changed r0 to 10
No operation
Final result: r0 = 10
```