컴퓨터구조론 hw2

2024 . 05 . 28 제출

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

```
#include <stdio.h>
#include <stdlib.h>
```

1. Introduction (과제 개요)

이번 과제는 부동 소수점 연산을 제외한 MIPS CPU 에뮬레이터를 만드는 것이었습니다. 에뮬레이터는 메모리와 프로그램을 바이너리 파일에서 로드하고, 예외 처리를 적절히 수행하며, PC가 0xFFFFFFFF에 도달하면 실행을 완료해야 합니다. 프로그램은 0x0에서 시작하고 스택 포인터는 0x1000000으로 초기화했습니다.

2. Background (배경 지식)

MIPS는 RISC(Reduced Instruction Set Computer) 아키텍처입니다. MIPS 명령어 실행의 다섯 단계(Fetch, Decode, Execute, Memory Access, Write back)를 이해하는 것이 에뮬레이터 구현에 필요한 배경지식이었습니다. 또한, MIPS 명령어 형식(R형, I형, J형)을 이해하고 이를 디코드하는 방법을 아는 것이 중요했습니다. 그리고 예외처리에 대한 여러 가지 방법들도 찾아봐야 했습니다.

3. Design (디자인)

프로그램은 모듈화를 염두에 두고 코드를 짰으며, MIPS 파이프라인의 단계를 함수로 분리했습니다.
-Fetch : 프로그램 카운터(PC)를 사용하여 메모리에서 명령어를 가져옵니다.
-Decode : 가져온 명령어를 디코드하여 그 유형과 피연산자를 식별합니다.
-Execute : ALU를 사용하여 명령어를 실행하고, 점프 명령어의 경우 PC를 업데이트합니다.
-Memory Access : I형 명령어에 대한 메모리 로드 및 저장 작업을 처리합니다.
-Write back : 결과를 적절한 레지스터에 씁니다.

4. Implementation (구현)

구현에는 MIPS 파이프라인의 다섯 단계가 모두 포함됩니다. 구현한 부분과 구현하지 않은 부분을 표로 정리하고 설명을 추가했습니다.

기능	구현 여부(O/X)	설명
Fetch	o	PC를 기반으로 메모리에서 명령어를 가져옵니다
Decode	o	R형, I형, J형 명령어를 디코드합니다
Execute	o	산술 및 논리 연산을 수행하고, 점프를 처리합니다
Memory Access	o	로드 및 저장 명령어를 처리합니다
Write back	o	결과를 적절한 레지스터에 씁니다
부동 소수점 연산	x	
예외처리	x	부적절한 메모리 액세스 및 지원되지 않는 명령어를 처리

Test case1

```
koicloud@ca-32234743:~/vscode$ ./32234743 input1/sum.bin
f0ffbd27> Cycle: 1
[Instruction Fetch] 0xf0ffbd27 (PC=0x00000000)
[Instruction Decode] Type: I, Inst: I, imm: bd27
opcode: 3c, rs: 07 (00000000), rt: 1f (ffffffff), rd: 00 (00000000), shamt: 00, funct: 00
RegDst: x, RegWrite: x, ALUSrc: x, PCSrc: x, MemRead: x, MemWrite: x, MemtoReg: x, ALUOp: xx
[Execute]
[Memory Access]
[Write Back] Target: rt, Value: 0x00000000 / newPC: 0x00000004
0c00beaf> Cycle: 2
[Instruction Fetch] 0x0c00beaf (PC=0x00000004)
[Instruction Decode] Type: J, Inst: J, address: 0000beaf
opcode: 03, rs: 00 (00000000), rt: 00 (00000000), rd: 00 (00000000), shamt: 00, funct: 00
RegDst: x, RegWrite: x, ALUSrc: x, PCSrc: x, MemRead: x, MemWrite: x, MemtoReg: x, ALUOp: xx
[Execute] Jump and Link to 0x0002fabc
[Memory Access] Pass
[Write Back] newPC: 0x0002fabc
NOP 명령어를 만났습니다. 프로그램을 종료합니다.
0002fac0> Final Result
Cycles: 3, R-type instructions: 0, I-type instructions: 1, J-type instructions: 1
Return value (v0): 0x0
```

Test case2

```
koicloud@ca-32234743:~/vscode$ ./32234743 input2/func.bin
d8ffbd27> Cycle: 1
[Instruction Fetch] 0xd8ffbd27 (PC=0x00000000)
[Instruction Decode] Type: I, Inst: I, imm: bd27
opcode: 36, rs: 07 (00000000), rt: 1f (ffffffff), rd: 00 (00000000), shamt: 00, funct: 00
RegDst: x, RegWrite: x, ALUSrc: x, PCSrc: x, MemRead: x, MemWrite: x, MemtoReg: x, ALUOp: xx
[Execute]
[Memory Access]
[Write Back] Target: rt, Value: 0x00000000 / newPC: 0x00000004
2408bfaf> Cycle: 2
[Instruction Fetch] 0x2408bfaf (PC=0x00000004)
[Instruction Decode] Type: I, Inst: I, imm: bfaf
opcode: 09, rs: 00 (00000000), rt: 00 (00000000), rd: 00 (00000000), shamt: 00, funct: 00
RegDst: x, RegWrite: x, ALUSrc: x, PCSrc: x, MemRead: x, MemWrite: x, MemtoReg: x, ALUOp: xx
[Execute]
[Memory Access]
[Write Back] Target: rt, Value: 0x00000000 / newPC: 0x00000008
2000beaf> Cycle: 3
[Instruction Fetch] 0x2000beaf (PC=0x00000008)
[Instruction Decode] Type: I, Inst: I, imm: beaf
opcode: 08, rs: 00 (00000000), rt: 00 (00000000), rd: 00 (00000000), shamt: 00, funct: 00
RegDst: x, RegWrite: x, ALUSrc: x, PCSrc: x, MemRead: x, MemWrite: x, MemtoReg: x, ALUOp: xx
[Execute] ALU result: 0xffffbeaf
[Memory Access]
[Write Back] Target: rt, Value: 0xffffbeaf / newPC: 0x0000000c
25f0a003> Cycle: 4
[Instruction Fetch] 0x25f0a003 (PC=0x0000000c)
[Instruction Decode] Type: I, Inst: I, imm: a003
opcode: 09, rs: 0f (00000000), rt: 10 (00000000), rd: 00 (ffffbeaf), shamt: 00, funct: 00
RegDst: x, RegWrite: x, ALUSrc: x, PCSrc: x, MemRead: x, MemWrite: x, MemtoReg: x, ALUOp: xx
[Execute] ALU result: 0xffffbeaf
[Memory Access]
[Write Back] Target: rt, Value: 0x00000000 / newPC: 0x00000010
04000224> Cycle: 5
[Instruction Fetch] 0x04000224 (PC=0x00000010)
[Instruction Decode] Type: I, Inst: I, imm: 0224
opcode: 01, rs: 00 (ffffbeaf), rt: 00 (ffffbeaf), rd: 00 (ffffbeaf), shamt: 00, funct: 00
RegDst: x, RegWrite: x, ALUSrc: x, PCSrc: x, MemRead: x, MemWrite: x, MemtoReg: x, ALUOp: xx
[Execute] ALU result: 0xffffbeaf
[Memory Access]
[Write Back] Target: rt, Value: 0x00000000 / newPC: 0x00000014
1c00c2af> Cycle: 6
[Instruction Fetch] 0x1c00c2af (PC=0x00000014)
[Instruction Decode] Type: I, Inst: I, imm: c2af
opcode: 07, rs: 00 (00000000), rt: 00 (00000000), rd: 00 (00000000), shamt: 00, funct: 00
RegDst: x, RegWrite: x, ALUSrc: x, PCSrc: x, MemRead: x, MemWrite: x, MemtoReg: x, ALUOp: xx
[Execute] ALU result: 0xffffbeaf
[Memory Access]
[Write Back] Target: rt, Value: 0x00000000 / newPC: 0x00000018
1c00c48f> Cycle: 7
[Instruction Fetch] 0x1c00c48f (PC=0x00000018)
[Instruction Decode] Type: I, Inst: I, imm: c48f
opcode: 07, rs: 00 (00000000), rt: 00 (00000000), rd: 00 (00000000), shamt: 00, funct: 00
RegDst: x, RegWrite: x, ALUSrc: x, PCSrc: x, MemRead: x, MemWrite: x, MemtoReg: x, ALUOp: xx
[Execute] ALU result: 0xffffbeaf
[Memory Access]
[Write Back] Target: rt, Value: 0x00000000 / newPC: 0x0000001c
0f00000c> Cycle: 8
[Instruction Fetch] 0x0f00000c (PC=0x0000001c)
[Instruction Decode] Type: J, Inst: J, address: 0300000c
opcode: 03, rs: 00 (00000000), rt: 00 (00000000), rd: 00 (00000000), shamt: 00, funct: 00
RegDst: x, RegWrite: x, ALUSrc: x, PCSrc: x, MemRead: x, MemWrite: x, MemtoReg: x, ALUOp: xx
[Execute] Jump and Link to 0x0c000030
[Memory Access] Pass
[Write Back] newPC: 0x0c000030
PC가 메모리 범위를 벗어났습니다: 0x0c000030
0c000030> Final Result
Cycles: 8, R-type instructions: 0, I-type instructions: 7, J-type instructions: 1
Return value (v0): 0x0
```

Test case3

```
koicloud@ca-32234743:~/vscode$ ./32234743 input3/fibonacci.bin
e0ffbd27> Cycle: 1
[Instruction Fetch] 0xe0ffbd27 (PC=0x00000000)
[Instruction Decode] Type: I, Inst: I, imm: bd27
opcode: 38, rs: 07 (00000000), rt: 1f (ffffffff), rd: 00 (00000000), shamt: 00, funct: 00
RegDst: x, RegWrite: x, ALUSrc: x, PCSrc: x, MemRead: x, MemWrite: x, MemtoReg: x, ALUOp: xx
[Execute]
[Memory Access]
[Write Back] Target: rt, Value: 0x00000000 / newPC: 0x00000004
1c00bfaf> Cycle: 2
[Instruction Fetch] 0x1c00bfaf (PC=0x00000004)
[Instruction Decode] Type: I, Inst: I, imm: bfaf
opcode: 07, rs: 00 (00000000), rt: 00 (00000000), rd: 00 (00000000), shamt: 00, funct: 00
RegDst: x, RegWrite: x, ALUSrc: x, PCSrc: x, MemRead: x, MemWrite: x, MemtoReg: x, ALUOp: xx
[Execute]
[Memory Access]
[Write Back] Target: rt, Value: 0x00000000 / newPC: 0x00000008
1800beaf> Cycle: 3
[Instruction Fetch] 0x1800beaf (PC=0x00000008)
[Instruction Decode] Type: I, Inst: I, imm: beaf
opcode: 06, rs: 00 (00000000), rt: 00 (00000000), rd: 00 (00000000), shamt: 00, funct: 00
RegDst: x, RegWrite: x, ALUSrc: x, PCSrc: x, MemRead: x, MemWrite: x, MemtoReg: x, ALUOp: xx
[Execute]
[Memory Access]
[Write Back] Target: rt, Value: 0x00000000 / newPC: 0x0000000c
25f0a003> Cycle: 4
[Instruction Fetch] 0x25f0a003 (PC=0x0000000c)
[Instruction Decode] Type: I, Inst: I, imm: a003
opcode: 09, rs: 0f (00000000), rt: 10 (00000000), rd: 00 (00000000), shamt: 00, funct: 00
RegDst: x, RegWrite: x, ALUSrc: x, PCSrc: x, MemRead: x, MemWrite: x, MemtoReg: x, ALUOp: xx
[Execute]
[Memory Access]
[Write Back] Target: rt, Value: 0x00000000 / newPC: 0x00000010
0f000424> Cycle: 5
[Instruction Fetch] 0x0f000424 (PC=0x00000010)
[Instruction Decode] Type: J, Inst: J, address: 03000424
opcode: 03, rs: 00 (00000000), rt: 00 (00000000), rd: 00 (00000000), shamt: 00, funct: 00
RegDst: x, RegWrite: x, ALUSrc: x, PCSrc: x, MemRead: x, MemWrite: x, MemtoReg: x, ALUOp: xx
[Execute] Jump and Link to 0x0c001090
[Memory Access] Pass
[Write Back] newPC: 0x0c001090
PC가 메모리 범위를 벗어났습니다: 0x0c001090
0c001090> Final Result
Cycles: 5, R-type instructions: 0, I-type instructions: 4, J-type instructions: 1
Return value (v0): 0x0
```

Test case4

```
koicloud@ca-32234743:~/vscode$ ./32234743 input4/factorial.bin
e0ffbd27> Cycle: 1
[Instruction Fetch] 0xe0ffbd27 (PC=0x00000000)
[Instruction Decode] Type: I, Inst: I, imm: bd27
opcode: 38, rs: 07 (00000000), rt: 1f (ffffffff), rd: 00 (00000000), shamt: 00, funct: 00
RegDst: x, RegWrite: x, ALUSrc: x, PCSrc: x, MemRead: x, MemWrite: x, MemtoReg: x, ALUOp: xx
[Execute]
[Memory Access]
[Write Back] Target: rt, Value: 0x00000000 / newPC: 0x00000004
1c00bfaf> Cycle: 2
[Instruction Fetch] 0x1c00bfaf (PC=0x00000004)
[Instruction Decode] Type: I, Inst: I, imm: bfaf
opcode: 07, rs: 00 (00000000), rt: 00 (00000000), rd: 00 (00000000), shamt: 00, funct: 00
RegDst: x, RegWrite: x, ALUSrc: x, PCSrc: x, MemRead: x, MemWrite: x, MemtoReg: x, ALUOp: xx
[Execute]
[Memory Access]
[Write Back] Target: rt, Value: 0x00000000 / newPC: 0x00000008
1800beaf> Cycle: 3
[Instruction Fetch] 0x1800beaf (PC=0x00000008)
[Instruction Decode] Type: I, Inst: I, imm: beaf
opcode: 06, rs: 00 (00000000), rt: 00 (00000000), rd: 00 (00000000), shamt: 00, funct: 00
RegDst: x, RegWrite: x, ALUSrc: x, PCSrc: x, MemRead: x, MemWrite: x, MemtoReg: x, ALUOp: xx
[Execute]
[Memory Access]
[Write Back] Target: rt, Value: 0x00000000 / newPC: 0x0000000c
25f0a003> Cycle: 4
[Instruction Fetch] 0x25f0a003 (PC=0x0000000c)
[Instruction Decode] Type: I, Inst: I, imm: a003
opcode: 09, rs: 0f (00000000), rt: 10 (00000000), rd: 00 (00000000), shamt: 00, funct: 00
RegDst: x, RegWrite: x, ALUSrc: x, PCSrc: x, MemRead: x, MemWrite: x, MemtoReg: x, ALUOp: xx
[Execute]
[Memory Access]
[Write Back] Target: rt, Value: 0x00000000 / newPC: 0x00000010
05000424> Cycle: 5
[Instruction Fetch] 0x05000424 (PC=0x00000010)
[Instruction Decode] Type: I, Inst: I, imm: 0424
opcode: 01, rs: 08 (00000000), rt: 00 (00000000), rd: 00 (00000000), shamt: 00, funct: 00
RegDst: x, RegWrite: x, ALUSrc: x, PCSrc: x, MemRead: x, MemWrite: x, MemtoReg: x, ALUOp: xx
[Execute]
[Memory Access]
[Write Back] Target: rt, Value: 0x00000000 / newPC: 0x00000014
0d00000c> Cycle: 6
[Instruction Fetch] 0x0d00000c (PC=0x00000014)
[Instruction Decode] Type: J, Inst: J, address: 0100000c
opcode: 03, rs: 00 (00000000), rt: 00 (00000000), rd: 00 (00000000), shamt: 00, funct: 00
RegDst: x, RegWrite: x, ALUSrc: x, PCSrc: x, MemRead: x, MemWrite: x, MemtoReg: x, ALUOp: xx
[Execute] Jump and Link to 0x04000030
[Memory Access] Pass
[Write Back] newPC: 0x04000030
PC가 메모리 범위를 벗어났습니다.: 0x04000030
04000030> Final Result
Cycles: 6, R-type instructions: 0, I-type instructions: 5, J-type instructions: 1
Return value (v0): 0x0
```

Test case5

```
koicloud@ca-32234743:~/vscode$ ./32234743 input5/power.bin
e0ffbd27> Cycle: 1
[Instruction Fetch] 0xe0ffbd27 (PC=0x00000000)
[Instruction Decode] Type: I, Inst: I, imm: bd27
opcode: 38, rs: 07 (00000000), rt: 1f (ffffffff), rd: 00 (00000000), shamt: 00, funct: 00
RegDst: x, RegWrite: x, ALUSrc: x, PCSrc: x, MemRead: x, MemWrite: x, MemtoReg: x, ALUOp: xx
[Execute]
[Memory Access]
[Write Back] Target: rt, Value: 0x00000000 / newPC: 0x00000004
1c00bfaf> Cycle: 2
[Instruction Fetch] 0x1c00bfaf (PC=0x00000004)
[Instruction Decode] Type: I, Inst: I, imm: bfaf
opcode: 07, rs: 00 (00000000), rt: 00 (00000000), rd: 00 (00000000), shamt: 00, funct: 00
RegDst: x, RegWrite: x, ALUSrc: x, PCSrc: x, MemRead: x, MemWrite: x, MemtoReg: x, ALUOp: xx
[Execute]
[Memory Access]
[Write Back] Target: rt, Value: 0x00000000 / newPC: 0x00000008
1800beaf> Cycle: 3
[Instruction Fetch] 0x1800beaf (PC=0x00000008)
[Instruction Decode] Type: I, Inst: I, imm: beaf
opcode: 06, rs: 00 (00000000), rt: 00 (00000000), rd: 00 (00000000), shamt: 00, funct: 00
RegDst: x, RegWrite: x, ALUSrc: x, PCSrc: x, MemRead: x, MemWrite: x, MemtoReg: x, ALUOp: xx
[Execute]
[Memory Access]
[Write Back] Target: rt, Value: 0x00000000 / newPC: 0x0000000c
25f0a003> Cycle: 4
[Instruction Fetch] 0x25f0a003 (PC=0x0000000c)
[Instruction Decode] Type: I, Inst: I, imm: a003
opcode: 09, rs: 0f (00000000), rt: 10 (00000000), rd: 00 (00000000), shamt: 00, funct: 00
RegDst: x, RegWrite: x, ALUSrc: x, PCSrc: x, MemRead: x, MemWrite: x, MemtoReg: x, ALUOp: xx
[Execute]
[Memory Access]
[Write Back] Target: rt, Value: 0x00000000 / newPC: 0x00000010
03000524> Cycle: 5
[Instruction Fetch] 0x03000524 (PC=0x00000010)
[Instruction Decode] Type: R, Inst: R, funct: 24
opcode: 00, rs: 18 (00000000), rt: 00 (00000000), rd: 00 (00000000), shamt: 14, funct: 24
RegDst: x, RegWrite: x, ALUSrc: x, PCSrc: x, MemRead: x, MemWrite: x, MemtoReg: x, ALUOp: xx
[Execute]
[Memory Access] Pass
[Write Back] Target: rd, Value: 0x00000000 / newPC: 0x00000014
0a000424> Cycle: 6
[Instruction Fetch] 0x0a000424 (PC=0x00000014)
[Instruction Decode] Type: J, Inst: J, address: 02000424
opcode: 02, rs: 00 (00000000), rt: 00 (00000000), rd: 00 (00000000), shamt: 00, funct: 00
RegDst: x, RegWrite: x, ALUSrc: x, PCSrc: x, MemRead: x, MemWrite: x, MemtoReg: x, ALUOp: xx
[Execute] Jump to 0x08001090
[Memory Access] Pass
[Write Back] newPC: 0x08001090
PC가 메모리 범위를 벗어났습니다: 0x08001090
08001090> Final Result
Cycles: 6, R-type instructions: 1, I-type instructions: 4, J-type instructions: 1
Return value (v0): 0x0
```

Lesson(맺음말)

타입별로 Decode하는 것 실패하였고,
각각의 return value값도 제대로 표시되지않았습니다
예외처리 하지 못하였고
적절한 십진수값도 자리에 넣지 못하였습니다
아마 레지스터에 저장된 값을 잘 못 불러왔거나
명령어 디코딩 및 실행 중의 주소 계산과 점프를 정확하게 처리하지 못한 것 같습니다. 무한루프도 난감한 문제중 하나였습니다.