

# Lab2 - Single-Cycle CPU

team21: 20200001조은국, 20200431박현빈

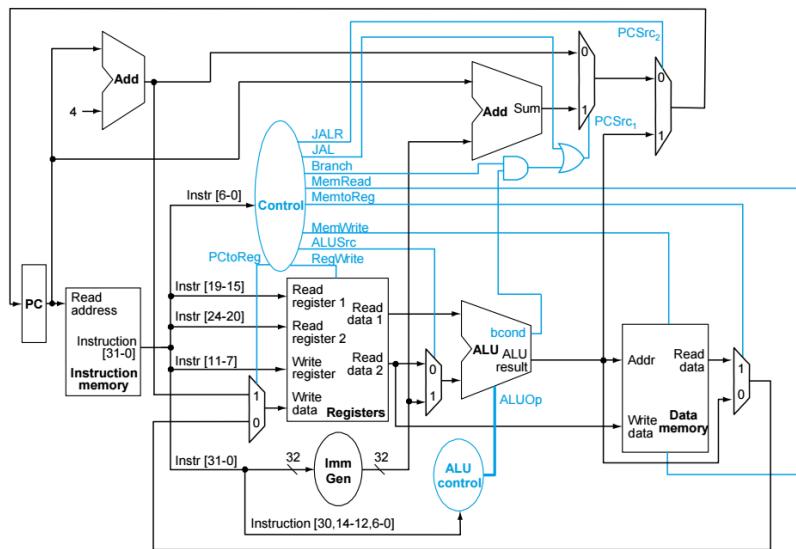
2022년 3월 29일  
(CSED 311)

## Introduction

본 Lab은 Single-Cycle RISC-V CPU를 구현하는 프로그램을 제작하는 과제이다. PC에서 명령어를 읽어와, 한 clock당 명령어에 맞는 하나의 명령어를 해결하도록 구현한다.

## Design

## Datapath with Control



34 POSTECH

위의 single cycle cpu에 대한 datapath와 control flow를 verilog를 통해 모방하는 방식으로 design하였다.

## PC (control.v)

input: next\_pc

output: current\_pc

다음 instruction의 address를 cpu의 계산 결과로부터 입력 받아 출력 한다.

## Instruction Memory(memory.v)

input: address

output: instruction

pc를 address로 받아 그에 해당하는 instruction을 출력 한다.

\*instruction fetch stage 담당

## Data Memory(memory.v)

input: address, data input / mem\_read, mem\_write => R/W를 정하는 control bit

output: data output

control에 따라 address에 해당하는 data memory의 값을 출력하거나, 해당 address의 memory에 특정 register의 값을 write한다.

\*memory access stage 담당

## Register(RegisterFile.v)

input: register 1,2, register direction, rd\_din / write\_enable

output: rs1\_dout, rs2\_dout

입력으로 받은 register에 해당하는 값을 출력하거나, control에 따라 register에 입력으로 받은 값을 write한다.

\*instruction decode stage, write back stage 담당

### **ALU control(operator.v)**

input: part\_of\_inst

output: alu\_op

instruction의 일부를 입력으로 받아 alu에서 수행해야 하는 연산에 대한 control을 출력한다.

Clock Asynchronous

### **ALU(operator.v)**

input: alu\_op, alu\_in\_1, alu\_in\_2

output: alu\_result, alu\_bcond

control에 따라 입력받은 두 값에 대한 연산을 수행하고, 연산 결과를 출력한다.

\*execute stage 담당

### **MUX(operator.v)**

input: control, input1, input2

output: out

control에 따라 입력받은 두 값 중 하나만을 출력한다.

### **Immediate generator(control.v)**

input: instruction[31:0]

output: immediate value

instruction에서 immediate부분을 도출하고, sign-extension하여 반환한다.

### **Control Unit(control.v)**

input: Opcode

output: 각각의 control signal bit (0 or 1)

opcode에 따라 여러 모듈(MUX, reg, mem 등)에 wire되는 control bit를 통제한다.

\*instruction decode stage 담당

이 외에도 덧셈 연산만 하는 add(Clock asynchronous)가 design되어 있다.

## Implementation

### PC

Current PC를 next\_PC로, Clock synchronous하게 update한다.

reset == 1일 경우 0으로 초기화

### Instruction Memory

PC address에 해당하는 instruction을 output으로 반환하도록, wire를 assign한다.

reset 시 memory를 초기화하고 파일을 읽어오는 기능을 수행한다.

Clock asynchronous

### Data Memory

input address에 해당하는 값을 반환하거나, rd에 저장한다.

Read는 posedge에 따라 Clock asynchronous, Write는 Clock synchronous

### Register

input에 해당하는 register 값을 반환하거나, rd에 저장한다.

Read는 posedge에 따라 Clock asynchronous, Write는 Clock synchronous

$rf[17] == 10$  일 시 is\_ecall을 위한 신호를 반환한다(Asynchronous).

## Control Unit

opcode에 맞게 Control Signal을 제어 한다. (Asynchronous)

CSED311 Lec5 lecture note 35-36 page 참고, if문으로 제작한다.

## ALU Control Unit

opcode, funct7, funct3 에 맞게 Control Signal을 제어 한다. (Asynchronous)

CSED311 Lec5 lecture note 35-36 page 참고, if문으로 제작한다.

## Immediate generator

if문으로 Opcode에 따라 immediate value part를 찾는다.

sign extension을 위해, 최고차항의 비트를 반복하여 32 bit를 채운다. 가령 12 bit imm\_value의 최고차항이 1인 경우, 앞의 20 bit를 1로 채운다.

## Clock asynchronous

## ALU

alu\_op에 따라 and, sub, or, xor, sll, srl을 계산하여 alu\_result로 반환한다

또는 alu\_op에 따라 eq, ne, lt, ge를 계산하여 alu\_bcond로 반환한다.

eq, ne, lt, ge 계산에는 SF와 OF라는 condition code를 활용한다.

## Clock asynchronous

## MUX

다른 모듈 구현 보조를 위해 구현한다.

## Discussion

각 모듈의 input, output이, 모듈간의 정보의 교환이 어떻게 연결되어 발생하는지 처음에 파악하는데 난항을 겪었다. 또한 각각의 모듈의 clock synchronous/asynchronous 구별이 혼란스러웠다. 수업자료를 비롯한 여러 자료를 찾아보며 토론하여 해결하였고, Single-Cycle CPU를 구조적으로 익힐 수 있었다.

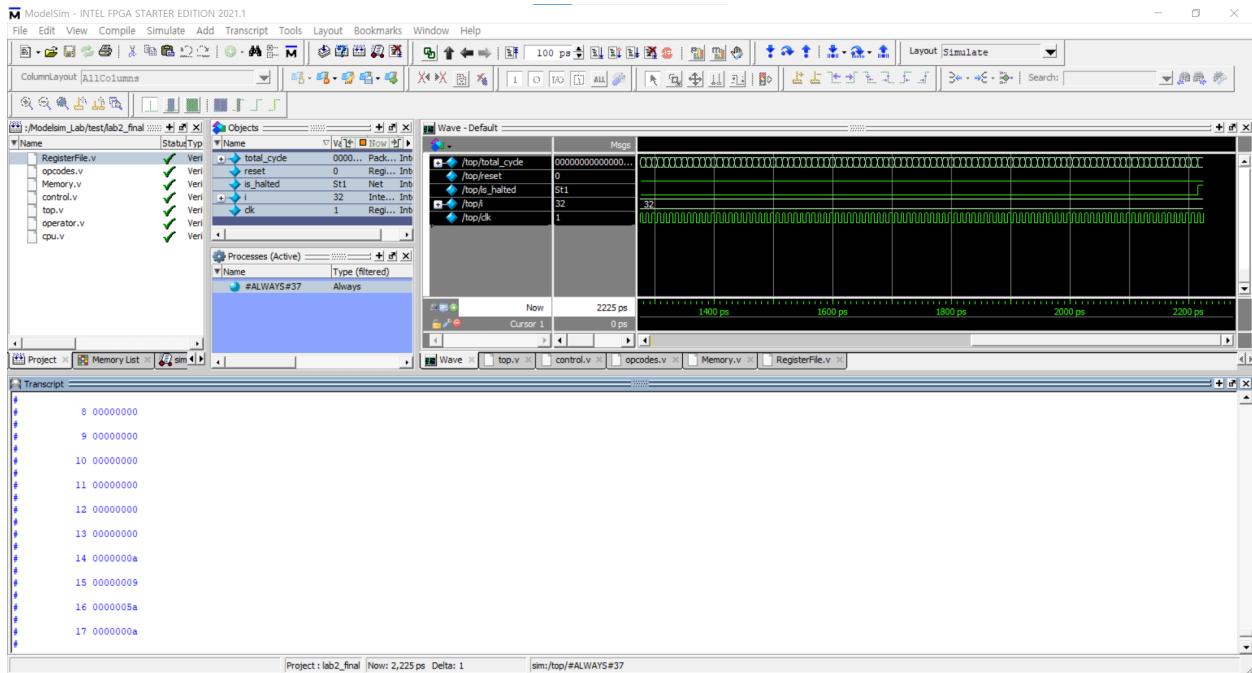
simulate 과정에서 modelsim exiting with error code 7이라는 warning이 뜨고 run이 되지 않는 문제가 발생했었는데, 같은 module에 대해 다른 방식으로 2번 declaration한 것이 원인이 된 error였다. 하나를 delete하자 문제가 해결되었다.

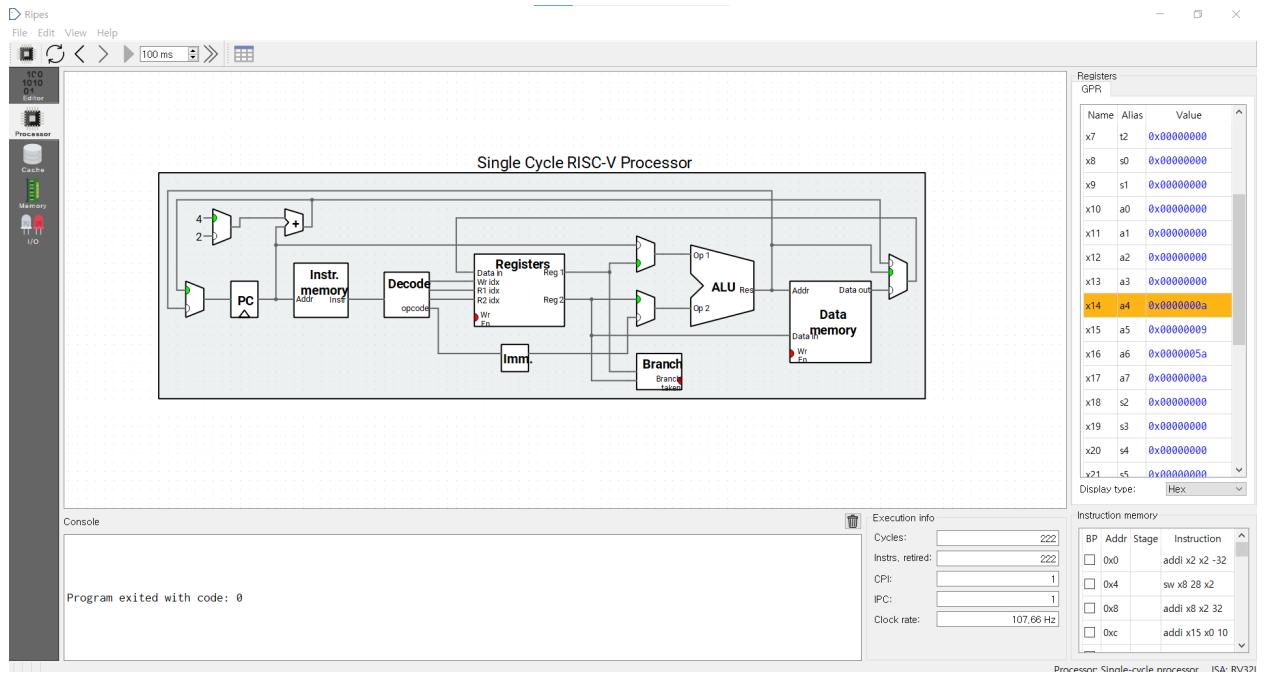
immediate-value의 bit를 쪼개고, sign-extension하는 부분에 실수가 더러 있었다.

## Conclusion

### 1. 실험 결과

여러 testbench code로 ripes의 결과와 비교해 보았을 때, 구현한 CPU가 정상적으로 작동함을 확인할 수 있다.





## 2. 결론

이번 Lab-Session에서는 모든 instruction을 동일한 한 clock에 처리하는 Single-Cycle CPU를 구현하였다. 한 Clock에 각 instruction의 stage가 처리되는 방식과 part들의 연결, 동작하는지 공부할 수 있었다. 각 module을 구현하며 clock synchronous와 clock asynchronous의 개념을 이해하고, verilog 문법으로 표현하는 방법도 익혔다.