**2024 Spring CSED311 Lab 3 Report**

Team ID: 67735

20220312 박준혁, 20220871 홍지우

**명예서약 (Honor Code)**

나는 이 프로그래밍 과제를 다른 사람의 부적절한 도움 없이 완수하였습니다.

I completed this programming task without the improper help of others.

1. **Introduction**

각 스테이지별로 쪼개어 연산하는 Multi-Cycle CPU를 Finite State Machine 기반으로 구현한다.

각 연산별로 요구 사이클이 다르기 때문에 빠른 연산은 빨리 끝낼 수 있다;  
오래 걸리는 연산에 사이클을 맞추던 싱글사이클보다 빠르게 동작한다.

1. **Design**
   1. 전체 구조

텍스트, 도표, 평면도, 개략도이(가) 표시된 사진

자동 생성된 설명

기존 싱글 사이클에서 A,B, inst, mem. aluout 레지스터, mux들이 추가된 형태이다.

* 1. 모듈 동기화 여부

Synchronous:

PC, Register File, Memory, IR, ALUOut, MDR, Control Unit

Asynchronous

ALU, ALU Control, ImmGen  
 Asynchronous read from memory and register

* 1. FSM Design

도표, 그림, 스케치, 원이(가) 표시된 사진

자동 생성된 설명  
IF를 두단계, EX를 두단계, MEM을 한단계로 구현하였다.  
EX2의 경우 branch taken을 처리하는 용도로 추가하였다.   
나머지 타입에 대한 연산은 EX1에서 끝난다

* 1. Microcode Control Diagram

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| State | Flow | R/I | LD | SD | Bxx | JALR | JAL |
| IF1 | next | - | - | - | - | - | - |
| IF2 | go to | ID | ID | ID | ID | ID | EX1 |
| ID | next |  |  |  |  |  |  |
| EX1 | go to | WB | MEM | MEM | EX2 | WB | WB |
| EX2 | go to |  |  |  | IF1 |  |  |
| MEM | go to |  | WB | IF1 |  |  |  |
| WB | go to | IF1 | IF1 |  |  | IF1 | IF1 |
| CPI |  | 5 | 6 | 5 | 5 | 5 | 4 |

1. **Implementation**
   1. Control Unit
      1. State Transition (mictrocode controller)

텍스트, 스크린샷, 소프트웨어, 폰트이(가) 표시된 사진

자동 생성된 설명 텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

* + 1. Module Port Declaration, Synchronization

텍스트, 스크린샷, 폰트, 소프트웨어이(가) 표시된 사진

자동 생성된 설명 텍스트, 스크린샷, 폰트, 디자인이(가) 표시된 사진

자동 생성된 설명

* + 1. Setting Control Values: Init, IF1, IF2

텍스트, 스크린샷, 폰트, 디자인이(가) 표시된 사진

자동 생성된 설명 텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명  
`ALU 시그널은 기존과 동일하게 다양한 연산을 하도록 한다.  
`ALU\_ADD, ALU\_SUB는 ALU가 더하기와 빼기만 하도록 한다.  
`ALU\_NOP는 아무런 결과도 생성하지 않도록 막는 역할을 한다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명 텍스트, 스크린샷, 폰트, 소프트웨어이(가) 표시된 사진

자동 생성된 설명  
  
텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명  
ID, EX 부분

Branch의 경우 EX1에서 조건을 계산하고, 만약 not taken이면 값을 쓰고 IF로 돌아간다.

만약 taken이면 EX2로 넘어가 PC=PC+imm을 하고 IF로 돌아간다.

Branch를 제외한 다른 연산은 EX1에서 연산이 끝난다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명  
MEM 부분: Load, Store의 경우에만 작동한다.  
Store의 경우에는 WB를 거치지 않고 여기서 끝난다.

텍스트, 스크린샷, 폰트, 소프트웨어이(가) 표시된 사진

자동 생성된 설명 텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명  
WB 부분  
각 Type에 맞는 PC 값을 업데이트 해 준다.

1. **Discussion**
   1. 논의
      1. Branch의 계산과 PC를 컨트롤 하는 것이 복잡했다.
      2. state를 여러 개로 쪼갰더니 레지스터에서 값이 사라지거나 하는 문제가 있어, 꼭 나누어야 하는 state를 제외하고는 하나로 합쳤다
      3. ALU가 원하지 않는 타이밍에 동작하는 것을 막기 위해 ALU\_NOP 시그널을 주어 계산하지 않도록 하였다.
2. **Conclusion**
   1. tb 결과

텍스트, 스크린샷이(가) 표시된 사진

자동 생성된 설명 텍스트, 스크린샷이(가) 표시된 사진

자동 생성된 설명 텍스트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

basic\_mem ifelse\_mem loop\_mem

텍스트, 스크린샷이(가) 표시된 사진

자동 생성된 설명 텍스트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

non-controlflow\_mem recursive\_mem

전부 동일한 레지스터 값을 얻을 수 있었다.