2025 Spring

CSED311

컴퓨터구조

Lab 4-2 report

Team ID: 15

팀원 1: 20230345 이성재

팀원 2: 20230355 정지성

목차

1. 서론
2. 디자인
3. 구현
4. 논의 사항
5. 결론

Report (Deadline: 5/13 18:00)

• How to handle branch prediction?

• Describe your design of branch predictor

• If you implement 2-bit global prediction

• Compare total cycles of 2-bit global prediction with that of always-taken and always-not-taken

• If you implement always-taken

• Compare total cycles of always-taken with that of always-not-taken

1. **서론**

이 과제의 목표는 Lab 4-1 에서 구현했던 pipelined cpu를 data hazard 뿐만 아니라 control hazard 도 해결할 수 있도록 변형하는 것이다. 또한 추가 점수를 위해, Branch prediction도 구현을 해야 하고 우리는 Gshare를 만들었다. 이 보고서에서 우리는 먼저 branch prediction을 하지 않은, 즉 always-not-taken 하는 cpu를 구현한 방법과 결과를 다루고 Gshare를 적용한 cpu에 대한 설명을 이후에 다루도록 하겠다.

1. **디자인**
2. **Always-not-taken CPU**

우리는 먼저 control flow instructions, 즉 BEQ, BNE, BLT, BGE, JAL, JALR 명령어들에 대해서 파이프라인이 정상적으로 작동하도록 always-not-taken 방식을 구현했다. IF 단계에서 읽은 명령어가 control instruction이더라도 항상 다음 pc를 pc+4로 예측하는 방식인데, 이는 매번 control instruction 인지 stall로 확인한 후 나아가는 방식보다는 높은 성능을 보이지만, 잘못 예측한 경우 flush 해야 하는 명령어들이 생긴다는 문제점이 있다. 이 과제에서는 특히 EX 단계에서 control flow를 잘못 예측했는지 확인해 잘못 예측한 경우 2개의 버블만 발생하도록 요구한다.

1. **구현**
2. **논의 사항**
3. **결론**