Single Cycle MicroArchitecture for MIPS

과목: 컴퓨터구조론

소속학과: 컴퓨터공학과

학번: 32184787

이름: 표세훈

제출일: 5월 11일

Deadline: 5월 13일

Free-Day: 1(22-05-18: 코드 오류 수정)

기본적 과제 설명

컴퓨터는 폰 노이만의 구조에 의해 구성되어 메모리에 프로그램이 올라가고 CPU는 PC 값에 따라 메모리에서 값을 읽어 와 분석 후 실행한 뒤 PC 값을 증가시켜 다음 메모리에서 값을 읽어 오는 방식을 채택하고 있다 이것이 Stored Program, Sequential Execution이다. 메모리로부터 값을 읽어온 CPU는 값을 분석 후 실행하는데 메모리로부터 데이터를 읽어 오는 단계를 Fetch Stage, 가져온 데이터에서 상위 6비트를 잘라 opcode를 뽑아내고 opcode에 맞추어 필요한 데이터를 뽑아내거나 데이터를 뽑아내 레지스터로 값을 읽어 오는 단계가 Decode Stage, 연산이 필요한 경우 값끼리 더하는 등 연산을 하는 단계가 Execution Stage, 필요한 경우 메모리로부터 값을 읽어오거나 메모리에 값을 쓰는 단계가 Memory Access Stage, 연산 된 값이나 메모리로부터 읽어온 값을 다시 레지스터로 쓰는 단계인 Write-Back Stage로 구성이 되어 이를 마치면 다시 Fetch 단계로 복귀한다. 이번 과제는 이러한 CPU의 구조를 파악 후 bin으로 작성된 프로그램 파일을 읽어 소프트웨어로 CPU를 구현해 보는 것이다.

구조 설명

본 과제를 수행하기 위해 다음 단계를 거쳐서 실행을 해야 한다. 1) bin으로 된 파일을 이진파일로 읽기와 동시에 메모리 상 한 줄 씩 입력하는 행위, 2) 메모리에 저장된 명령어를 PC값에 의해 한 줄 읽어오는 단계 (Fetch Stage), 3) PC 값을 +4 하는 단계 (update pc) 4) 읽어온 명령어를 각기 opcode, rs, rt, rd … 등으로 나눠서 구조체에 입력하는 단계와 만약 jump instruction이면 pc = R[rs]으로 바로 이동하는 단계(Decode Stage), 5) opcode에 의해 필요한 계산을 하는 단계 (Execute Stage), 6) 필요한 경우 메모리에 Write 혹은 Read 하는 단계 (Memory Access Stage), 7) 필요한 경우 계산한 값이나 메모리로부터 읽어온 값을 다시 레지스터에 입력하는 단계 (Write Back Stage) 의 단계를 거쳐 다시 1)으로 돌아가며, 만약 pc가 -1이면 계산을 끝내고 (루프 종료) 계산한 결과를 출력하는 것으로 프로그램이 종료된다. 이 과정에서 필요한 구조를 집어보자면, 프로그램과 값을 저장할 메모리, pc 값, 명령어를 분석해 저장할 구조체가 기본적으로 필요하며 추가적으로 cycle 수, R, I, J type 실행 수 등을 체크할 변수가 부가적으로 필요하다. 본 과제를 수행함에 있어서 다소 헤맸던 부분이 존재한다. 수업 시간 중 본 과제를 시작하기 위한 기초 환경의 구성을 실습으로 진행하였으나, 운영체제의 다름으로 인해 windows에서 실행을 하려고 하니 ntohl가 실행이 되지 않았다. 이를 해결하기 위해 #include <Winsock2.h>, #pragma comment(lib,"ws2\_32")를 추가해 해결하였다. 그 다음 문제는 decode 단계에서 일어났다. 명령어를 shift 연산을 이용해 분리하는 것 까지는 할 수 있었다. 다만 그 값을 main 함수로 다시 돌아오는 과정에서 프로그램이 종료되는 현상이 발생하였다. 이는 decode 함수에서 return ret을 하지 않아 발생하는 것으로 수정한 결과 실행할 수 있었다. 그 다음은 Memory Access Stage에서 일어났다 메모리에 접근해 값을 입력 혹은 출력하는 단계에서 에러가 발생이 되었다. 그 결과 Memory Access Address에 문제가 있음을 발견할 수 있었다. 하나의 예시로 MIPS Green Sheet에 의하면 M[R[rs]+SignExtImm] = R[rt]의 구조로 구현을 했어야 하나 그 당시에는 M[rs + SignExImm] = R[rt]의 구조로 구현을 하여 잘못된 주소로 접근을 하고 있었다. 이를 수정하니 정상적으로 동작하였다. 오류도 존재하였지만 의문점도 몇 개 존재하였다. 먼저 Jump Instruction의 구현 방식 문제였다. 직접 짠 구조에서 Decode에서 각기 값으로 분리 후 Execute에서 switch 문을 이용해 opcode로 계산을 하는 단계였다. 초기 구조에선 이 구조를 이용해 jump 할 위치를 파악 후 write back 단계에서 switch 문을 이용한 opcode 분기, 값을 반환해 보다 깔끔하게 구현을 할 수 있었다. 그러나 실제 컴퓨터는 Jump Instruction은 Fetch 후 Decode 하는 2단계로 종료된다고 수업시간에서 배웠다. 그 점에서 의문을 가져 인터넷 검색을 해 보았으나 신뢰할 수 없는 사이트 중 많은 사람들이 쓰는 사이트인 wiki에서 다른 것은 다 맞게 되어있었으나 jump의 경우 3 단계로 되어 있었다. 수업시간에 배운 내용과 괴리가 생겨 더 검색을 시도했으나 왜 그렇게 적혀 있는지는 밝혀 낼 수 없었다. 그래서 추후 수정을 통해 decode 단계에서 jump의 경우 종료되게 재 구성하였다. 다른 의문점은 ‘input4.bin’를 처리함에 있어서 발생하였다. Input4.bin의 경우 구조가 1만개의 데이터를 메모리에 저장한 후 sort 과정을 통해 정렬 후 최종적으로 101번째 값을 사용자에게 반환하는 프로그램이다. 이 과정에서 sort 프로그램이 삽입 정렬이라 시간 복잡도가 n2의 형태를 가져 다른 프로그램들은 20초 내로 끝났으나 input4.bin 만 3시간 정도 걸려 프로그램이 종료되었다. 보다 가벼운 sort 프로그램을 제작하기 위해 교수님의 도움을 받아 bin 파일 만드는 방법을 배워 만든 프로그램을 실행을 시도하였으나, 직접 만든 프로그램은 돌아가지 않는 결과가 발생하였다. 의문이 들어 input4.c 파일과 input\_data.h 파일을 그대로 사용해 bin 파일을 제작해 비교를 한 결과 같은 코드에서 생성한 bin이 서로 코드로 작성이 되었음을 발견할 수 있었다. 교수님이 제시한 과정을 그대로 수행하였으나 같은 파일에 대해 다른 코드가 작성되었는지 의문이 든다. 교수님과 상의한 결과 컴파일 프로그램이 다르다는 것을 알게 되었고 다른 컴파일을 찾았으나 mips-linux-gnu-gcc를 설치하는 예제만 있어 다른 방법을 찾지 못하였다.

22년 5월 18일 추가. 금일 강의에서 코드리뷰 중 메모리 관련 부분에서 오류를 발견하여 Free Day를 사용해 코드 수정을 하였다. 기존 코드에서는 Execution 단계에서 나온 value를 memory 배열의 주소로 사용하였는데 이 과정에서 value 값을 그대로 주소로 사용한 점이 오류였다. 실제로 구현을 하기 위해선 value에서 나온 주소를 memory배열의 주소로 바로 들어가는 것이 아닌 value/4를 해 이를 주소로 사용해야 한다. 예제 코드에서는 오류가 발견이 안 돼서 테스트 환경에서 오류를 발견하지 못해 넘어간 점으로 보인다. Memory 범위는 16M으로 설정되어 데이터 사이에 3칸씩 띄워서 입력이 되고 출력을 해도 문제가 없는 것처럼 보였기 때문에 예제를 돌리는데 있어 오류점을 발견하지 못한 것으로 추측된다.

컴파일 환경

Visual Studio 2022 & Windows 10 x64bit

돌아가는 예제

‘simple.bin’ – 프로그램 시작 & 종료

텍스트이(가) 표시된 사진

자동 생성된 설명

‘simple2.bin’ – 100을 int 변수에 할당 후 반환텍스트이(가) 표시된 사진

자동 생성된 설명

‘simple3.bin’ – 0 부터 100까지 더한 후 반환텍스트이(가) 표시된 사진

자동 생성된 설명

‘simple4.bin’ – 재귀함수를 이용한 1 부터 10까지 더한 후 반환텍스트이(가) 표시된 사진

자동 생성된 설명

‘fib.bin’ – 재귀함수를 이용한 10번째 피보나치 수열 결과 반환텍스트이(가) 표시된 사진

자동 생성된 설명

‘gcd.bin’ - 0x1298와 0x09387의 최대공약수 구한 후 반환텍스트이(가) 표시된 사진

자동 생성된 설명

‘input4.bin’ - 1만개의 데이터를 입력 후 sort 과정을 거친 후 101번째 작은 수 반환

텍스트이(가) 표시된 사진

자동 생성된 설명

‘fib\_fix.bin’ – ‘fib.bin’ 을 기반으로 jal -> jalr 코드로 수정텍스트이(가) 표시된 사진

자동 생성된 설명

과제 하며 배운 것

이번 과제가 CPU의 구조를 직접 코드로 작성하는 것이 목표였다. 수행하기 전에는 각 단계가 무엇을 수행하는지 모호하던 상태에서 과제 수행을 하면서 각 단계가 무엇을 하는지 분명히 할 수 있다는 점에서 본디 목적을 이룰 수 있었다. 과제에서 벗어나서 느낀 것은 현재 사용중인 CPU의 Multi Cycle이랑 Pipeline이 Single Cycle에 비해 얼마나 높은 성능을 보이는 지였다. Input4.bin의 경우 수행을 3시간에 걸쳐서 나온 결과가 일치하는지 직접 판단하기가 어려웠다. 그래서 따로 프로젝트를 만든 후 Input4.c 파일을 붙여 넣은 후 실행을 시켜 보니 바로 결과가 나왔다. 당시 결론이 일치하는 것에 집중을 먼저 한 것 보다는 3시간에 걸쳐 실행한 프로그램이 단지 순간적으로 결과가 나왔다는 것에 놀랐다. 다른 것은 이번 과제를 하면서 기존 서버 돌리는 용도로 방치하고 있던 리눅스를 보다 활용하는 능력을 기를 수 있었다는 점이다. 리눅스 ssh 접속부터 파일 및 폴더 생성/삭제, vim 사용법, scp 파일 보내기/받기, gcc를 이용한 리눅스 상 c언어 코딩, 등 활용 방법을 배울 수 있었으며 추후 본 과제를 리눅스로 돌리는 시도를 해 볼 예정이다. 또한 본 과제를 git과 github을 이용해 git 사용의 연습으로 삼을 수 있는 기회가 되었다.

Git Repository

https://github.com/kimpyo9357/Computer-Instruction/settings