어셈블리프로그래밍설계및실습 보고서

실험제목: Second operand & Multiplication

실험일자: 2016년 10월 06일 (목)

제출일자: 2016년 10월 12일 (수)

학 과: 컴퓨터공학과

담당교수: 이형근 교수님

실습분반: 화5, 목6,7

학 번: 2013722095

성 명: 최 재 은

1. 제목 및 목적
   1. 제목

Second operand & Multiplication

* 1. 목적

Bit stream을 움직여 값을 곱하거나 나누는 LSL, LSR을 이용하여 10까지의 팩토리얼 값을 구하고 이를 MUL opcode를 이용하여 구현해 본 뒤 두 코드의 차이를 비교해 본다. 이를

비교해 봄으로써 어떤 방식의 스코어가 더 좋은지를 확인해본다.

1. 설계 (Design)
   1. Pseudo code
      * 1. **Second operand**

R4에 temp 주소 값을 저장

R3에 1을 저장(1!), r4 주소 번지에 저장하고 r4값에 4 더함

R3를 왼쪽으로 1bit 이동(2!)하여 r0에 저장

r4 주소 번지에 저장하고 r4값에 4 더함

r0와 r0를 왼쪽으로 1bit 움직인 값의 합을 r1에 저장(3!)

r4 주소 번지에 저장하고 r4값에 4 더함

r0에 r1을 왼쪽으로 2bit움직인 값을 저장(4!)

r4 주소 번지에 저장하고 r4값에 4 더함

r0와 r0를 왼쪽으로 2bit 움직인 값의 합을 r1에 저장(5!)

r4 주소 번지에 저장하고 r4값에 4 더함

r2에 r1을 왼쪽으로 1bit이동시킨값을 저장

r0에 r2와 r1을 왼쪽으로 2bit 이동시킨 값의 합을 저장(6!)

r4 주소 번지에 저장하고 r4값에 4 더함

r1에 RSB를 이용하여 r0를 왼쪽으로 3bit움직인 값에서 r0를 빼준 값을 저장(7!)

r4 주소 번지에 저장하고 r4값에 4 더함

r0에 r1을 왼쪽으로 3bit 이동시킨 값을 저장(8!)

r4 주소 번지에 저장하고 r4값에 4 더함

r1에 (r0 + r0를 왼쪽으로 3bit이동시킨 값) 을 저장(9!)

r2에 r1을 왼쪽으로 1bit이동시킨 값을 저장

r0에 r2와 r1을 왼쪽으로 3bit이동시킨 값의 합을 저장(10!)

* + - 1. **Multiplication**

R2에 1저장

R1에 2저장

R3에 temp 주소 값 저장

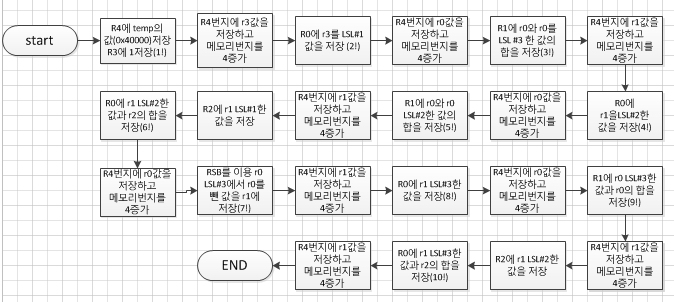
Loop선언

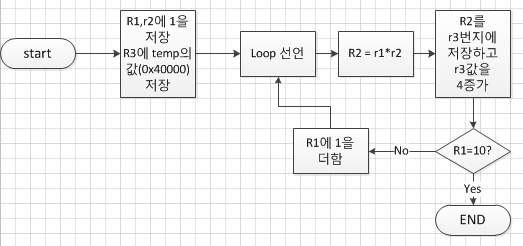
R2에 r1\*r2값을 저장

R2값을 r3메모리 번지에 저장하고 r3값을 4 증가

R1이 10인지 확인

아니라면 r1을 1 증가시키고 loop로 점프

* 1. Flow chart 작성
     + 1. Second operand
       2. Multiplication



* 1. Result

**1) Second operand**



- 1! ~ 10!까지의 값이 little endian방식으로 저장되어있음을 확인.

**2) Multiplication**



* Second operand에서와 동일한 값이 저장되어있음을 확인할 수 있음.
  1. Performance
     + 1. Second operand

C:\Users\Home\Desktop\sec size.JPG

C:\Users\Home\Desktop\sec state.JPG

- code size는 100이고 state는 35개입니다. Score는 code size \* state^2

이므로 이 프로그램의 스코어는 12250이 나오게 됩니다.

* + - 1. Multiplication

C:\Users\Home\Desktop\mul size.JPG

C:\Users\Home\Desktop\mul state.JPG

- code size는 40이고 state는 84개입니다. Score는 code size \* state^2

이므로 이 프로그램의 스코어는 282240이 나오게 됩니다.

1. 고찰 및 결론
   1. 고찰 및 결론

처음에 10!만을 구하는 줄 알고 2의 제곱수들의 합을 통해서 구했다가 모두 구해야 된다는 말을 듣고 급하게 수정했습니다. 심지어 실수로 짰던 것이 코드 길이도 어마어마 했던 것을 생각하니 웃음밖에 안 나옵니다….. 뭐 아무튼 수업시간 때 실습했던 내용을 이용하여 최대한 코드 길이를 줄여보았지만 일일이 저장해줘야 하는 것 때문에 state가 급증한 것을 확인했지만 그래도 큰 어려움은 없었습니다. 두 개의 방식으로 동일한 결과를 도출하였지만 second operand는 코드길이가 긴 대신 state가 적었고, Multiplication 방식은 코드가 짧은 대신에 state가 훨씬 많았습니다. 스코어를 비교해보면 second operand가 Multiplication의 절반도 안 되는 것을 확인할 수 있습니다.

이번 과제를 진행하면서 뼈저리게 느낀 것은 사람에게 편하게 작성하면 스코어가 커지고,

스코어를 생각하면 사람이 힘들어진다는 것이었습니다.

4. 참고문헌