2021-1컴퓨터구조 HW1

201602955 바이오 메디컬 공학부 4학년 장현준

담당 교수: 고석훈 교수님

2021-1

1. **runProgram에 대한 구현 소개**

**runProgram에는 code\_bgn이 들어가게 됩니다. 그러면 매개변수로 넘어오게 된 UINT addr에는 메모리에 있는것 처럼 실제 실행시켜 할 명령 구분과 명령을 시킬 3자리의 주소가 같이 넘어오게 됩니다. 그럼 반복문을 코딩하기 시작하는데 전체적인 그림은 code\_bgn 부터 code\_end까지 진행하게 됩니다. 반목분은 for(int PC = addr; PC < code\_bgn;) 으로 진행 함으로써 명령어를 찾아서 fetch를 하면서 PC를 2씩 증가하는 식으로 코딩을 진행했습니다. 실제로 메모리에 들어가있는 알파벳을 관찰한 결과 소문자로 들어가있는 것을 확인했기에 PRT,PRC,PRS를 구분하기 위해서 대문자로 바꿔주는 과정이 필요했습니다. 그에 따라 아스키코드값을 이용해 대문자로 바꿔주는 알고리즘을 넣어주었습니다.**

**그 다음으로 제가 만든 RunProgram 에서의 핵심은 다음과 같습니다.**

1. **sprintf의 이용**

**sprintf를 이용해 16진수로 들어가있는 메모리에 1바이트씩 떨어져있는 2개의 16진수를 Instruction 이란 문자열 배열에 index [0] [1] [2] [3] 순서로 다루기 쉽게 문자열로 묶는 방법을 사용했습니다. 이렇게하면 [0] 을 바탕으로 이게 무슨 명령임을 뜻하는지 구분 할 수 있게 됩니다. 뿐만 아니라 이러한 방법으로 index [1] [2] [3] 을 이용하게 되면 다시 3자리 문자열로 묶을 수 있기에 명령어 와 그것을 실행해야하는 data영역으로 접근 할 수 있게 됩니다.**

1. **strtol 의 이용**

**아직 위에서 언급했듯이 3자리의 문자열이기 때문에 이것을 16진수로 바꿔주는 기술이 필요하게 됩니다. 그로 인해 사용한 것이 문자열을 16진수로 바꿔줄 수 있는 strtol 라이브러리 함수를 사용했습니다.**

**이와 같은 함수들을 이용해 명령어를 구분 할 수 있게 되고 data영역에 접근 할 수 있는 준비를 마치고 명령어에 따라 코딩을 진행했습니다.**

**먼저 연산을 하는 ADD,SUB,MUL,IAC의 경우 ACC값을 계속 연산해 가지만 연산 과정이 C에서 일어나야 하기 때문에**

sprintf(address,"%c%c%c",instruction[1],instruction[2],instruction[3]);

**다음과 같이 연산을 해야하는 숫자를 메모리번지를 묶고**

IR\_address = strtol(address,NULL,16);

**그걸 16진수로 바꾼 뒤**

int temp\_num = accnum2cint(readWord(IR\_address))

**계산하기전 값을 C로 불러오고**

c\_num = accnum2cint(ACC) + temp\_num;

**원래 있던 누산기 값을 C로 변환해서 불러오고 값을 계산하게 됩니다.**

ACC = cint2accnum(c\_num);

**C 값을 누산기 값에 넣게 됩니다.**

**이와 같은 방법으로 C에서 연산을 할 때 연산 부호만 다르게 했을 뿐 나머지 과정은 전부 동일 합니다.**

**LDA와 STA 연산을 할 때는 sprintf와 strtol을 쓰는 것은 동일 하지만 핵심 적인 코드 구현 부분은 다음과 같습니다.**

**LDA:**

c\_num = accnum2cint(readWord(IR\_address));  
ACC = cint2accnum(c\_num);

**IR\_address에는 load해야하는 메모리 번지가 있게 되고 그것을 잠시 c\_num에 저장한뒤 다시 그 값을 ACC에 넣게 됩니다.**

**STA:**

//이제 메모리를 바꿔야한다.  
writeWord(IR\_address,ACC);

**실질적인 메모리 값을 바꿔야하므로 제공 되어진 write word를 이용하도록 코드를 구현 했습니다.**

**HLT,IAC는 strcmp 문자열을 비교하는 함수(strcmp)를 이용하여 구현을 진행했습니다.**

**그리고 PRC,PRT,PRS는 이미 제공되어 있었기에 그함수에 IR\_address를 집어넣어 메모리 번지만 제공해주는 식으로 코드를 구현했습니다.**

1. **이차방정식 프로그램 구현**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

**이차방정식을 진행시키는 전체적인 프로그램은 다음과 같습니다.**

**Data section에서는 100~106번지 까지는 A,B,C,X를 할당 하고 112번에는 Y를 할당했습니다. 방정식을 저장하는 부분은 108번지 부터 110 까지 16진수로 기록 되어있습니다.**

**Code section 에서는 A를 먼저 Load하고 Load된 누산기 값에 X를 두번 곱하는 MUL을 하게 되고 그 값을 다시 Y에 저장한 뒤 B를 load 하여 다시 X를 곱하고 C를 더한 뒤 Y에 더하고 다시 Y에 저장하는 알고리즘으로 설계를 진행 했으며 이를 기계어로 변경하면 위 사진에서 볼 수 있듯이 다음과 같은 순서가 됩니다.**

1. **이차방정식 프로그램 테스트 결과**

**각각의 테스트 결과 사진을 보여주면 다음과 같습니다.**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

**첫번째 테스트 케이스 결과가 정상적으로 잘 출력한 것을 확인할 수 있습니다.**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

**두번째 테스트 케이스결과가 정상적으로 잘 출력한 것을 확인 할 수 있습니다.**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

**세번째 테스트 케이스 결과까지 잘 출력된 것을 확인 할 수 있습니다.**

**다음 사진은 4번째 테스트 케이스 결과와 5번째 테스트 케이스 결과입니다.**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

**두개의 테스트 케이스 결과는 값이 다르게 나온 것을 확인 하였는데 이는 Memory overflow가 발생 된 것을 알 수 있습니다. 그러면 왜 이런 현상이 벌어질 수 밖에 없는지에 대해 서술을 하면 이는 accnum2cint 나 cint2accnum 에있는 data 영역의 크기가 7FFF이기 때문입니다. 이를 10진수로 바꾸면 32767임을 알 수 있습니다. 위에서 언급한 두개의 함수에서 & 즉 비트 연산이 일어나게 되는데 이를 직접 계산한 사진은 다음과 같습니다.**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

**위 사진은 제가 직접 아이패드에서 비트 연산을 계산한 사진이며 4번째 테스트 케이스 에서 90 ^ 2 \* 50 의 연산이 일어나게 됩니다. 그러면 10진수 값은 40500이고 이를 비트연산을 위해 2진수로 변경하면 1001111000110100 이고 7FFF를 2진수로 변경하게 되면 0111111111111111 이렇게 변경이 되어 집니다. 이를 비트 연산을 수행하면 위의 제가 직접 계산한 결과가 나오게 되며 프로그램의 결과 오버플로우가 날 수 밖에 없는 구조였음을 확인 할 수 있게 됩니다. 5번째 테스트 케이스도 결국 음수의 범위가 -32367까지 이므로 5번째 테스트 케이스연산인 -50 \* 30 ^ 2 = - -45000임을알수 있고 이는 결국 4번째 테스트 케이스와 같은 오버 플로우가 일어난것을 확인 할 수 있습니다. 역시 비트 연산이 일어날 것이고 이는 결국 다른 값이 누산기에 저장됨을 알 수 있습니다.**