시스템 프로그래밍 개인 과제 보고서

-SIC 및 SIC/XE를 이용한 알고리즘 작성-

제출일 : 2019년 11월 1일 (금)
교수님 성함 : 고영배 교수님
학과 : 소프트웨어학과
학번 : 201720723
이름 : 박수린

1. 알고리즘에 대한 개요

이 프로그램의 목적은 입력 받은 정수에 대한 소인수 분해를 진행하는 것이다. 소인수분해는 합성수를 소수들의 곱으로 표현하는 것이다.

전체 프로그램은 크게 MAIN의 SCAN과 RESULT로 이루어져 있다. 또한 입출력을 위한 method로, INPUT과 OUTPUT을 사용하였다. 처음에 INPUT을 통해 첫 번째 줄의 숫자를 입력 받고 COUNT에 저장한 후 1~10까지의 숫자인지 유효성 검사를 거친다. 그 이후 워드 단위인 3을 카운트에 곱하여 인덱스 register인 X을 사용하여 3씩 증가시켜 count 개수만큼 수를 입력 받아 ARRAY에 저장한다.

그 후 결과를 출력하는 RESULT로 진행된다. RESULT에서 JSUB을 이용해 COUNT의 개수만큼 FACT을 호출한다. FACT에서는 먼저 return address을 저장하고 원래의 수를 SELF라는 변수에 저장해 둔다. 그 후, 본격적인 factorization이 수행된다. FACTOR라는 변수는 처음에 2로 초기화 되어 있고, SIC과 SIC/XE에는 사칙연산 외에 나머지 연산이 없기 때문에, FACTOR로 나눈 후에 다시 곱하였을 때 같은 값이면 FACTOR로 나누어 떨어지는 것으로 판단하였다. 나누어 떨어질 경우에는 FACTOR와 '*' 곱하기 기호를 출력한 후 다시 돌아가 그 몫을 FACTOR로 나누어 떨어지는 지 확인한다. (몫은 QUOTA라는 변수에 저장하였다.) 만약 나누어 떨어지지 않는다면 (FACTOR로 나눈 후 다시 곱했을 때 원래의 수와 결과가 다르다면) FACTOR을 1 증가 시킨 후 같은 작업을 반복한다.

그 다음은 소수의 판단이다. 몫을 FACTOR로 나누는 작업을 계속 하다가 몫이 1일 경우에는 소인수분해가 끝났다고 판단하였다. 만약 소인수분해가 끝났을 때의 FACTOR가 SELF에 저장해둔 처음 수와 같다면 소수라고 판단하고 P을 출력하였다.

처음 RESULT에서 FACT을 호출할 때 return address을 따로 저장한 이후는, FACT에서 반복 수행을 하면서 JSUB을 사용하기 때문이다. FACT 수행이 끝나면 RETURN에 저장해 두었던 return address을 다시 호출하여 STL을 통해 L register에 다시 저장해준 후, RSUB을 통해 RESULT로 돌아간다. RESULT가 끝나면 전체 프로그램이 종료된다.

2. 실행 결과 Screen Shot 및 결과에 대한 설명

✧ 올바른 input에 대한 실행 결과

```
surin@surin-VirtualBox:~/SicTools$ java -jar out/make/sictools.jar
5
50 9 71 4 10

2*5*5
3*3
P
2*2
2*5

3
12 37 997

2*2*3
P
P

```

입력과 출력 사이는 enter로 구분 하였으며, 첫 입력으로 5개의 숫자 50, 9, 71, 4, 10을 주었을 때 올바르게 출력되는 것을 알 수 있다. 두 번째 입력으로, 12 37 997의 3가지 숫자를 주었을 때 또한 올바르게 출력된다.

```
surin@surin-VirtualBox:~/SicTools$ java -jar out/make/sictools.jar
3
303 614 93

3*101
2*307
3*31

```

출력되는 숫자에 0이 있는 경우에도 올바르게 처리된다.

✧ 유효하지 않은 입력에 대한 예외처리

```
surin@surin-VirtualBox:~/SicTools$ java -jar out/make/sictools.jar
11
2 3 4 5 6 7 8 9 10 11 12
ERROR! Please Restart.

```

첫 번째 줄에 10보다 큰 정수가 입력되면 에러 메시지를 출력한다.

```
surin@surin-VirtualBox:~/SicTools$ java -jar out/make/sictools.jar
3
5 20 1000
ERROR! Please Restart.

```

두 번째 줄에 999보다 큰 정수가 입력되면 에러 메시지를 출력한다.

```
surin@surin-VirtualBox:~/SicTools$ java -jar out/make/sictools.jar
3
5 20 1
ERROR! Please Restart.

```

두 번째 줄에 2보다 작은 정수가 입력되면 에러 메시지를 출력한다.

```
surin@surin-VirtualBox:~/SicTools$ java -jar out/make/sictools.jar
3
10 20 3c
ERROR! Please Restart.
```

숫자가 아닌 올바르지 않은 입력에 대해서 에러 메시지를 출력한다.

3. 구현 사항 및 개선점

과제 요구 사항에 있는 것은 모두 구현하였다. 정수 배열이 주어졌을 때 숫자와 숫자 사이는 ENTER와 SPACE로 구분하여, 첫 번째 입력은 COUNT에 나머지 정수 배열은 ARRAY에 COUNT 개수 만큼 저장하였다.

특히 수를 입력 받을 때, INPUT이라는 subroutine을 사용하였다. Sictool의 입출력은 아스키 값으로 읽히기 때문에 ascii code에서 0을 의미하는 0x30을 빼서 숫자로 만든 후 저장하였다. 또한 여러 자리 수를 읽을 때가 구현이 어려웠다. 예를 들어 234를 입력했을 때, 2를 읽은 후 다음 input이 SPACE나 ENTER인지, 또는 유효하지 않은 Input인지 검사하였다. 유효한 input이라면 숫자라는 의미이므로 2에 10을 곱한 후, 3을 더하였다. 다음에도 같은 방법으로 유효성 검사를 거친 후 4의 입력에 대해 처음 23에 10을 곱한 후 4를 더하여 234를 읽을 수 있었다. 값을 읽고 숫자로 저장한 이후 subroutine을 종료하였다. 값의 범위에 검사는 첫 번째 줄과 두 번째 줄이 다르므로 각각의 subroutine이 종료 된 후 확인하였다.

output 또한 구현이 어려웠는데 input에서 사용한 방법 정반대로 구현하였다.

개선점에 대해서는 error 처리를 함에 있어서 예를 들어 3개를 입력한다고 하고, 4개를 입력하면 3개만 읽어서 factorization까지 잘 동작이 되지만 에러 메시지를 출력하지는 않는다. count*3 값을 T레지스터에 저장하여 인덱스 레지스터인 X와 비교하는데, JLT을 사용하여 작을 때까지만 입력을 받기 때문에 4번째 입력된 값은 아예 저장되지 않도록 처리했다.

ENTER와 SPACE 입력에 관해서도 개선점이 있다. 첫 번째 줄에 1~10 사이의 숫자를 입력한 이후, enter 입력 후 정수의 배열을 받아야 하는데, 과제 문서에서 수와 수 사이는 띄어쓰기로 구분한다는 것과 혼동되어 수와 수 사이를 enter 또는 space로 구분하는 것으로 구현하였다.

4. 고찰

이번 과제를 통해서 Sictool을 사용하는 방법에 대해서 알게 되고, 코드 한 줄 한 줄 실행할 때마다 변수 값이 변하는 것을 바로 알 수 있고, Register에 저장된 값을 알 수 있어서 디버깅이 매우 편하다는 생각을 하였다. sic와 sic/xe 명령어에 대해서도 많이 익숙해졌으며, sic/xe 버전에 대해서는 더 공부가 필요하다고 생각했다. 또한 어셈블리어로 직접 코드를 작성해본 적이 처음이기 때문에 코드 길이는 길어지지만 훨씬 직관적이고 생각보다 흥미로워서 얻어갈 것이 많은 과제였다고 생각된다.