

Ucode Interpreter 분석

2016112158 김희수



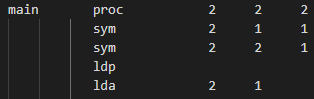
[날짜]

[회사 이름]

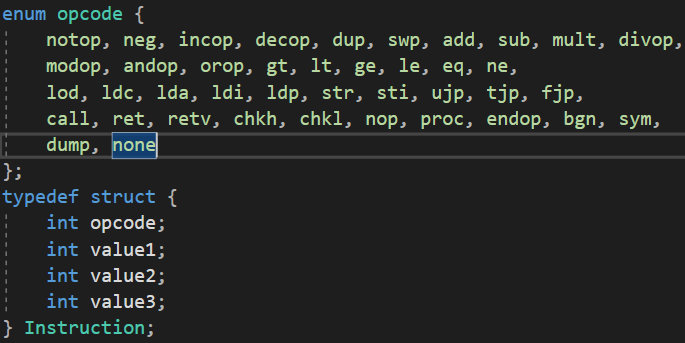
[회사 주소]

1. Ucode 인터프리터 구조분석

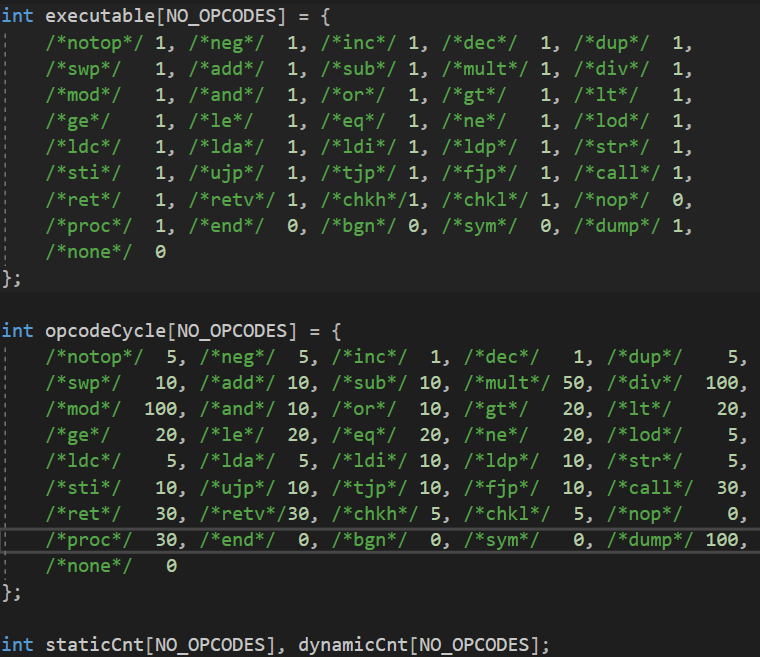
먼저 Ucode의 구조를 살펴보자



Ucode의 명령어 한 라인은 위와 같이 opcode value1, value2, value3의 포맷을 이룬다. 어셈블리어와 비슷하다. 이 명령어 한 라인은 소스코드 내에서 Instrucuion 구조체로 표현된다. opcode는 열거형으로 opcode의 이름은 문자열로 표현되었다.

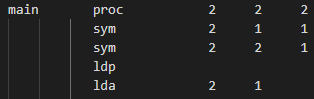
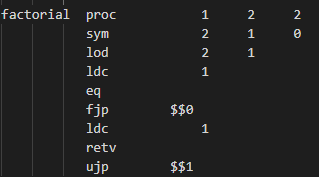


executable 배열과 opcodeCycle배열은 opcode의 실행여부와 실행될 때 소요되는 사이클수로 생각된다. 또한 staticCnt와 dynamicCnt는 opcode를 인덱스로 하여 각 opcode가 static, dynamic instruction에서 얼마나 사용되었는지 저장한다.



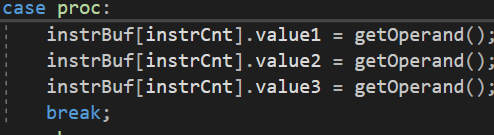
main 함수의 과정은 크게 두가지다. 입력으로 들어온 중간언어인 \*.uco파일을 object code로 바꾸는 assemble과정, 이 object code를 해석하여 실제로 실행시키는 execute과정이다. assemble과정은 Assemble클래스에 기술되어있고, execute과정은 Interpret 클래스에 기술되어있다.

먼저 Aseemble클래스부터 살펴보자. Assemble 클래스에서 가장 중요한 함수는 assemble함수이다. 다른 함수들은 assemble함수를 서포트하는 함수이다. 먼저 부차적인 함수들부터 살펴보고 assemble함수를 기술하겠다. getLabel함수는 Label을 가져오는 함수이다. Label은 Ucode에서 가장 처음으로 보이는것이라 보겠다. 밑의 사진에서 main, factorial에 해당하는거다. call이나 jump를 위한 subroutine name이라 볼 수 있다.



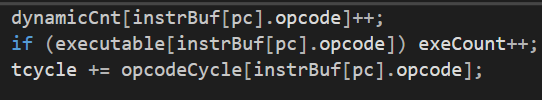
Label 역시 Label 클래스로 처리된다. Label클래스에는 Label을 labelTable에 삽입하는 insertLabel, labelTable에서 해당 Label을 찾는 findLabel, fix(명령어 주소와 다음 명령어 포인터)를 추가하는 addFix함수가 짜여져있다. 여기서 labelTable은 원소에 Label들의 이름과 주소, 그리고 명령어 리스트를 저장하는 배열이다.

다시 Assemble 클래스로 돌아가자. assemble()함수에서 Ucode의 라인들은 임시로 lineBuffer배열에 저장된다. getLabel()은 lineBuffer에서 해당라인의 Label을 가져오는 함수이고, getOpcode()는 해당라인의 lineBuffer에서 opcode를 가져오는 함수이다. getOperand()는 lineBuffer에서 operand를 가져오는 함수이다. 또한, Ucode를 한줄마다 읽어오므로 라인을 인덱싱하기 위한 instrCnt가 줄마다 증가된다. 이제 assemble 함수를 보겠다. assemble함수에서 Ucode에서 한줄마다 lineBuffer에 저장하고 getLabel()과 getOpcode()를 실행시킨다. 읽어온 opcode와 operand는 instrBuf에 저장된다. 또한, 해당 opcode를 인덱스로 하는 staticCnt를 증가시킨다. 이를 통해 staticCnt는 assemble과정에서 해당 opcode가 얼마나 사용되었는지가 저장되는 배열이라 할 수 있겠다. opcode로 case를 나누어 명령어를 실행시키는 것이다. main proc 2 2 2로 예를 들어보겠다. getLabel()을 통해 lineBuffer에서 main을 읽어와 insertLabel을 통해 labelTable에 저장될 것이다. 그 후, getOpcode()로 opcode를 읽어오고 현재 opcode는 proc이다. proc의 경우 다음과 같은 작업을 수행한다.



요약하자면 main proc 2 2 2의 경우, lineTable에 “main” Label이 저장되어있고, instrBuf[instrCnt](instrCnt는 0이다)에 저장되며 저장되는 opcode==proc이고, value1==2, value2==2,value3==2이다. assemble함수의 끝에는 instrWrite()함수가 실행된다. instrWrite함수는 instrBuf의 내용, 즉 object code를 출력파일에 write하는 과정이 담겨있다.

assemble이 완료되었으면 이제 interpret하여 실행할 차례이다. Interpret클래스에서 가장 중요한 부분은 execute()함수이다. execute()함수에서 object code의 시작주소를 받아서 시작주소부터 실행을 해나가는 것이다. 이 실행은 곧 연산을 의미하므로 연산을 수행하기 위한 스택이 있어야한다. 이 스택은 UcodeiStack로 선언되어있다. 즉, UcodeiStack클래스에는 스택에 대한 함수들(push, pop 같은)이 기술되어있다. object code의 시작주소를 pc로 하여 pc에 해당되는 opcode를 인덱스로 하는 dynamicCnt, exeCount, 그리고 실행되는데 소요되는 cycle수를 계산한다. dynamicCnt는 실행과정에서 해당 opcode가 얼마나 사용되었는지를 저장하고, opcode가 실행가능한지 확인하고 실행가능하면 실행횟수를 증가시킨다. 끝으로 opcode 연산에 소요되는 cycle수를 더해 총 소요되는 cycle수를 계산한다.

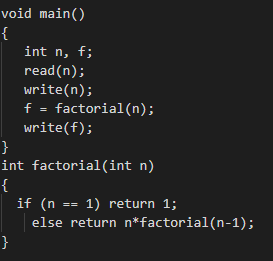


switch-case문을 통해 instrBuf의 opcode에 따라 수행되는 연산이 달라지도록 구현되어있다. 연산들은 UcodeiStack을 통해 이루어지는 스택연산이다. findAddr()함수는 주소연산을 할 때 사용되는 함수로 역할은 pc에 해당되는 메모리주소를 찾아준다. predefinedProc는 subroutine call이 일어날떄 그 call이 Label 클래스를 정의할 때 미리 정의해놓은 read,write,lf일 경우 그에 해당하는 처리를 하는 함수이다. execute()함수의 끝에는 statistic()함수가 실행되는데 이 함수는 object code 실행 후, assemble과정과 execute과정에서 사용된 opcode의 종류와 종류마다의 개수 , 전체 명령어 개수, 소요된 전체 사이클 수를 출력파일에 write해준다.

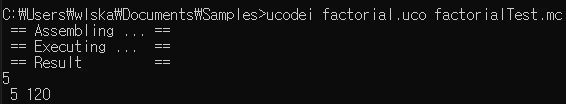
**2. Ucode 프로그램 실험 및 결과 분석**

1) factorial.uco

factorial.mc는 다음과 같다.

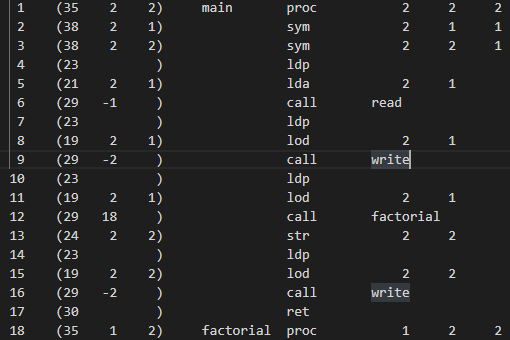


입력값으로 n을 받고 그 n을 인수로 factorial연산을 수행한 값을 f에 저장한 후, n과 f를 출력하는것이다. 즉 5를 입력하면 120이 출력되어야 한다(5! == 120)

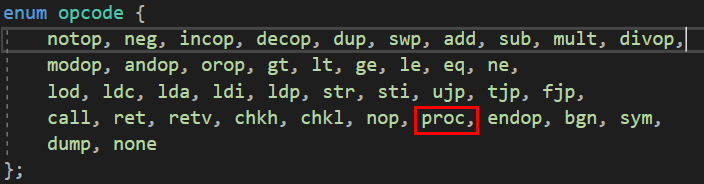


factorial.uco를 ucodei를 통해 실행시켰을 때, n에 5를 입력으로 주고 출력되는 n은 5, f는 120이므로 올바르게 실행됬음을 알 수 있다.

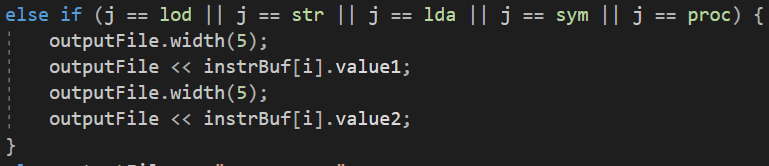
factorialTest.mc에는 ucodei를 실행시킨 후, object code와 ucode, 그리고 statistic들이 출력되었다



line을 보면 왼쪽은 object code, 오른쪽은 ucode이다. ucode상에서 main proc 2 2 2로 되어있는데 이때 proc는 opcode, 2 2 2는 operand들이다. 이게 object code에 대응되는지 확인해보자. 이를 확인하려면 Assemble클래스의 instrWrite()함수를 보면 알 수 있다. 먼저 proc는 35에 해당된다. 이는 열거형 opcode를 보면 알 수 있다. sym 2 1 1, sym 2 2 1은 순서대로 변수 n,f를 가리킨다

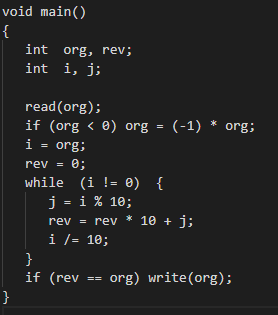
 proc는35번째 인덱스이다.

또한, object code (35 2 2)에서 2 2는 각각 ucode의 value1과 value2에 해당한다 이때 value1이 2인 이유는 value1 : (size of paras + sizeof local vars)로 factorial의 경우 n,f밖에 없기 떄문이다. value2는 block number로 함수에서 2가 된다.

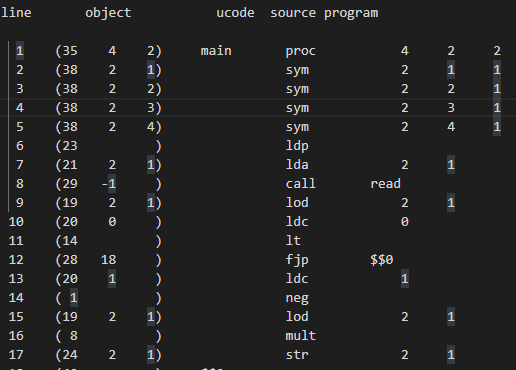
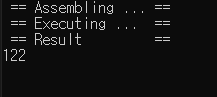
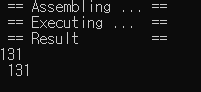


2) pal.uco

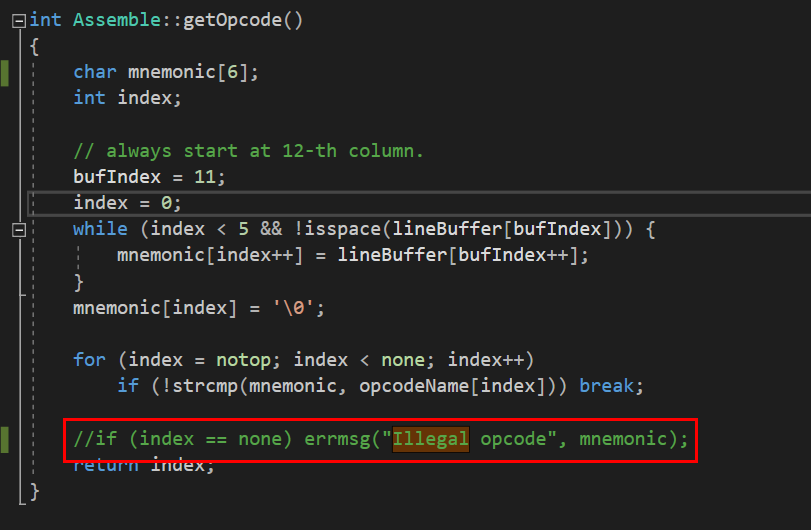
pal.mc는 다음과 같다.



org를 입력받아 paildrome number인지 확인하는 함수이다. 121, 1221이면 회문이므로 그대로 출력한다

처음에 주어진 pal.uco파일에서 line 7이 lda가 아니라 ldr로 되어있었다. lda로 바꿔주었다. sym 2 1 1 ~ sym 2 4 1은 순서대로 변수 org, rev I, j이다. 또한 마지막 $$3에서 ret end bgn이어야 하는데 가운데 end가 없었다. 다만 어째서인지 이 uco를 그대로 출력하면 illegal opcode라는 오류가 계속 떴었다. 왜인지 몰라 그냥 예외처리를 없애주었다.



**3. 홀수 마방진 출력**

C++로 나타낸 홀수마방진 출력은 다음과 같다.

#include <iostream>

using namespace std;

int main()

{

    int arr[225];

    int length, size, row, col, i;

    cin >> size;

    if (size % 2 == 0)

        return 0;

    else

    {

        if (size < 3)

            return 0;

        else

        {

            if (size > 15)

                return 0;

            else

            {

                col = size / 2;

                row = 0;

                length = i = size \* size;

                while (length > 0)

                {

                    length--;

                    int element = size \* row + col;

                    arr[element] = i - length;

                    if ((i - length) % size == 0)

                        row++;

                    else

                    {

                        row--;

                        col++;

                    }

                    if (row < 0)

                        row = size - 1;

                    if (col > (size - 1))

                        col = 0;

                }

                length = i;

                col = 0;

                while (length > 0)

                {

                    cout << arr[i - length] << " ";

                    length--;

                    col++;

                    if (col >= size)

                    {

                        col = 0;

                        cout << "\n";

                    }

                }

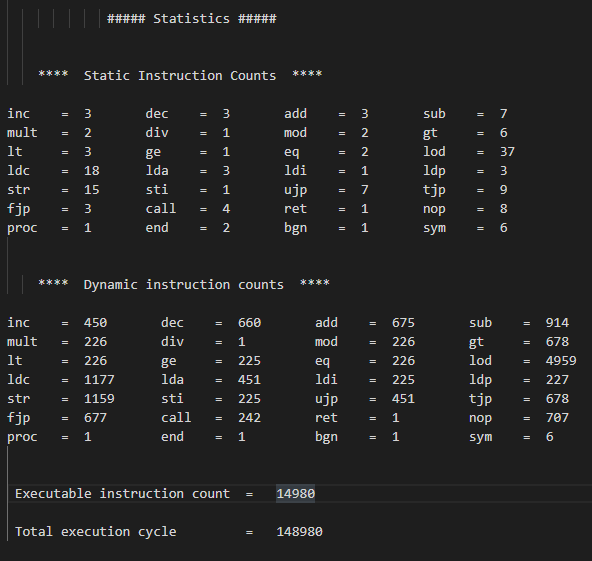
            }

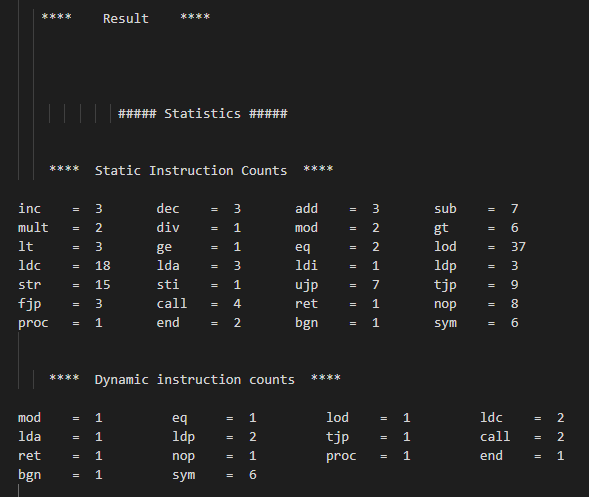
        }

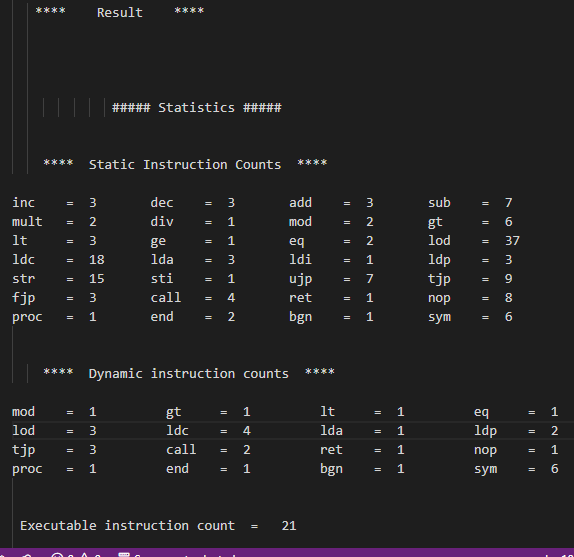
    }

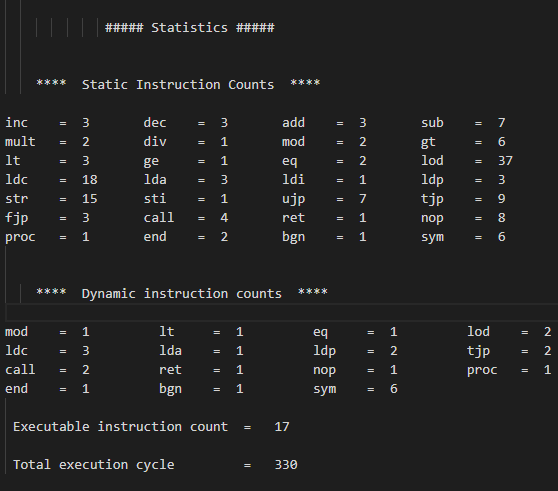
}

짝수가 입력되면 프로그램을 끝내고 3~15사이의 홀수가 입력되면 마방진을 생성한다. 3~15를 벗어나면 프로그램을 끝낸다

15를 입력한 경우

 6을 입력한 경우

 17을 입력한 경우

 1을 입력한 경우이다.

while문으로 재입력받는건 못하겠습니다.