어셈블리어 - 공학용 계산기 만들기 보고서

12171607 김한비

1. 구현 환경

언어: C++, 어셈블리어

프로그램: Visual Studio

2. 함수 구현 설명

◎ 연산함수

- Calc() 함수에서 mod기능과 ^기능을 구현하기 위해 필요한 double a = 0.49999, double b= 0 를 선언해준다.

1. 뺄셈

|  |  |
| --- | --- |
| \_\_asm{  fld fval1  fld fval2  fsub  fstp res  } | fsub:st(1)-st(0)을 하는 명령어  fval1,fval2를 로드해서 각각 st(1),st(0)에 저장한다.  fsub 이용해서 뺄셈을 한다.  st(0)에 저장된 값을 res에 리턴하고 pop한다 |

1. 곱셈

|  |  |
| --- | --- |
| \_\_asm{  fld fval1  fld fval2  fmul  fstp res  } | fmul:st(1)\*st(0)을 하는 명령어  fval1,fval2를 로드해서 각각 st(1),st(0)에 저장한다.  fmul 이용해서 곱셈을 한다.  st(0)에 저장된 값을 res에 리턴하고 pop한다 |

1. 나눗셈

|  |  |
| --- | --- |
| **\_\_asm{**  **fld fval1**  **fld fval2**  **fdiv**  **fstp res**  **}** | fdiv:st(1)/st(0)을 하는 명령어  fval1,fval2를 로드해서 각각 st(1),st(0)에 저장한다.  fdiv 이용해서 나눗셈을 한다.  st(0)에 저장된 값을 res에 리턴하고 pop한다 |

1. 제곱근

|  |  |
| --- | --- |
| \_\_asm{  fld fval  fsqrt  fstp res  } | fsqrt : 제곱근을 계산하는 명령어  fval을 로드해서 st(0)에 fval을 넣는다.  fsqrt를 사용해서 st(0)에 있는 fval의 제곱근을 구한다.  fstp를 사용해서 st(0)에 저장된 제곱근을 res에 리턴하고 st(0)에 있는 값을 pop한다. |

1. MOD

|  |  |
| --- | --- |
| \_\_asm{  fld fval1  fld fval2  fdiv  fld a  fsub  frndint  fld fval2  fmul  fld fval1  fsub st(0), st(1)  fstp res  } | 나머지 연산 논리 : a와 b를 나눈 몫을 구하고 a-b\*몫을 구하면 나머지를 구할 수 있다.  frndint : 정수로 반올림 하는 명령어, 근데 1.5는 반올림 하면 1이 된다.  fval1,fval2를 로드해서 각각 st(1),st(0)에 저장한다.  fdiv 이용해서 나눗셈을 한다. fdiv는 몫만 구하는게 아니라 소수점자리까지 나눗셈을 해준다.  정수부분 몫만 구하기 위해 나눗셈 값에서 소수점 부분을 버리기 위해 미리 선언한 a=0.49999를 fdiv한 값에서 빼준다.  그래서 나온 값을 frndint하면 버림이 되서 정수부분 몫을 구할 수 있다.  fval2와 정수부분 몫을 곱하고 fval1을 로드한다.  fval1-fval2\*정수부분 몫을 계산해서 나머지를 구한다.  st(0)에 저장된 값을 res에 리턴하고 pop한다. |

1. 2제곱

|  |  |
| --- | --- |
| \_\_asm{  fld fval  fmul fval  fstp res  } | fval을 로드해서 st(0)에 fval을 넣는다.  st(0)에 fmul을 사용해서 fval을 곱하고 st(0)에 곱한 값을 저장한다.  fstp를 사용해서 st(0)에 저장된 값을 res에 리턴하고 st(0)에 있는 값을 pop한다. |

1. n제곱

|  |  |
| --- | --- |
| \_\_asm{  fld fval2  fld fval1  fyl2x  fist b fild b  fsub  f2xm1 fld1  fadd  fild b fxch  fscale fstp res  } | n제곱 연산 논리: 로그함수를 이용, x^(a+b)=x^a+x^b 임을 이용  fyl2x: y\*log2x  fist: integer 저장  fild: integer 값 로드, integer 아니면 integer로 바꿔서 로드  f2xml: 2^x-1  fxchg: 레지스터 값을 서로 교환  fscale: ST(1)의 정수를 ST(0)의 지수에 추가.  fval2,fval1을 로드해서 각각 st(1),st(0)에 저장한다.  fy12x 이용해서 log2fval2^fval1 형태로 만들어준다.  st(0)에 있는 값이 0이 아닐 때 그 값을 부호 있는 정수로 바꿔주고 결과를 b에 저장한다.  (b에 log2fval2^fval1의 정수 부분 저장)  b를 로드한다.  st(1)-st(0)을 해준다, st(0)에는 소수 부분만 남음  f2xml 이용해서 2^(소수부분)-1을 st(0)에 저장한다.  fld1을 이용해 스택에 1을 푸쉬한다.  2^소수부분-1+1을 한다.  저장되어 있는 수 b를 로드한다.  fxch 이용해서 st(1)에 있는 2^소수부분과 st(0)에 있는 b를 서로 바꾼다.  fscale 이용해서 st(1)에 있는 정수 값 b를 st(0)의 지수에 추가한다.  따라서 fval2^fval1 값 출력 가능  st(0)에 저장된 값을 res에 리턴하고 pop한다. |

◎ 괄호 연산(구현하려다 실패)

|  |  |
| --- | --- |
| void CMFCApplication6Dlg::OnBnClickedButtonLp()  {  if (eflag == true)  m\_input = "";  eflag = false;  m\_input.AppendChar('(');  char vchar1[100];  char vchar2[100];  wcstombs(vchar1, m\_temp, 100); //m\_input 담는 배열 vchar1  wcstombs(vchar2, m\_op, 100); //연산자들 담는 배열 vchar2, 연산자들 따로 구분하려고 만듬  double fval1 = atof(vchar1);//vchar1에 있는 숫자를 double형으로 바꿈  double res = 0;  if (m\_op == "")  {  fval1 = 0;  m\_op = '+';  }  for (int i = 0; i < 100; i++)  {  if (m\_temp == '(')  break;    else  {  array[i] = vchar1[i];  symbol[i] = vchar2[i];  }  }    \_\_asm{    }  UpdateData(false);  }  void CMFCApplication6Dlg::OnBnClickedButtonRp()  {  if (eflag == true)  m\_input = "";  eflag = false;  m\_input.AppendChar(')');  char vchar3[100];  char vchar4[100];  char vchar5[100];  wcstombs(vchar3, m\_temp, 100); //m\_input 담는 배열 vchar1  wcstombs(vchar4, m\_op, 100); //연산자들 담는 배열 vchar2, 연산자들 따로 구분하려고 만듬  wcstombs(vchar5, m\_input, 100);  double fval1 = atof(vchar3);//vchar1에 있는 숫자를 double형으로 바꿈  double fval2 = array2[0];  double res = 0;    for (int i = 0; i < 100; i++)  {  if (m\_temp[i] == '(')  {  fval1 = 0;  m\_op = '+';  }  else if (m\_temp[i] == ')')  break;  else  {  array2[i] = vchar3[i];  symbol[i] = vchar4[i];  }  }  \_\_asm{  fld fval1  fld fval2  fstp res  }  UpdateData(false);  } | Wcstombs: M\_input이 wchar\_t형이기 때문에 배열에 넣고 자유롭게 쓸 수 있도록 사용한다.  Atof: Char형으로 값을 실수로 바꿔준다.  ‘(‘를 만나기 전 입력 받은 숫자들과 연산자들을 array와 symbol에 저장하고, ‘(’를 만나면 그 괄호 안에 있는 수를 새로운 배열 array2,symbol1에 넣는다. 인라인 어셈블리로 괄호 안의 값을 먼저 계산한 후 남아있는 나머지 값을 차례로 계산한다.  5+(3\*4)를 예시로 들어서 설명  5 : array에 들어감  +:symbol에 들어감  3,4: array2에 들어감  \*: symbol2에 들어감  배열에 넣은 걸 ) 부분 어셈블리에서 계산 |

◎ 사용 방법

- 실수 연산 가능, 음수 연산 가능,

- 뺄셈, 곱셈, 나눗셈, MOD, n제곱: 숫자, 연산자, 숫자 순으로 입력하고 = 누름.

- 루트: 루트 할 숫자 입력 후에 기호 입력

- 2제곱: 제곱하고 싶은 수 입력 후에 기호 입력

- n제곱 : 밑수,기호,지수 순으로 입력하고 = 누름

- 괄호 없이 세 개 이상의 수 누르면 앞에 두 개가 먼저 연산됨

◎ UI 디자인(계산기 배경색, 텍스트 색상들, 버튼 색상 변경)

- 계산기 배경색 변경

|  |  |
| --- | --- |
| BOOL CMFCApplication6Dlg::OnInitDialog()  {  CDialogEx::OnInitDialog();  SetBackgroundColor(RGB(218, 217, 255)); ASSERT((IDM\_ABOUTBOX & 0xFFF0) == IDM\_ABOUTBOX);  ASSERT(IDM\_ABOUTBOX < 0xF000);  CMenu\* pSysMenu = GetSystemMenu(FALSE);  if (pSysMenu != NULL)  {  BOOL bNameValid;  CString strAboutMenu;  bNameValid = strAboutMenu.LoadString(IDS\_ABOUTBOX);  ASSERT(bNameValid);  if (!strAboutMenu.IsEmpty())  {  pSysMenu->AppendMenu(MF\_SEPARATOR);  pSysMenu->AppendMenu(MF\_STRING, IDM\_ABOUTBOX, strAboutMenu);  }  }  SetIcon(m\_hIcon, TRUE);  SetIcon(m\_hIcon, FALSE);  return TRUE;  } | MFCApplication6Dlg.cpp의 93번째 함수 안에 setbackgroundcolor(color) 함수를 추가해 계산기 전체 배경 색을 변경했습니다. |

- 연산자 버튼들 텍스트 색상 변경

|  |  |
| --- | --- |
| dc.SetTextColor(RGB(255, 255, 255)); | OnDrawItem 함수에 setTextColor 함수 추가했습니다. |

- m\_op,m\_temp,m\_input 텍스트 색상 변경

|  |  |
| --- | --- |
| HBRUSH CMFCApplication6Dlg::OnCtlColor(CDC\* pDC, CWnd\* pWnd, UINT nCtlColor)  {  HBRUSH hbr = CDialogEx::OnCtlColor(pDC, pWnd, nCtlColor);  pDC->SetTextColor(RGB(255, 54, 181)); //메시지창 텍스트 색 변경  return hbr;  } | SetTextColor을 이용해 텍스트 색상 변경했습니다. |

- 숫자버튼(연산자 아닌 버튼들) 색상 변경

|  |  |
| --- | --- |
| 너무 많아서 코드는 몇 개만 가져왔어요  #define aa RGB(255,217,250)  #define bb RGB(255,108,235)  if (nIDCtl == IDC\_BUTTON2)  {  setBtnColor(&dc, rect, aa);  }  if (nIDCtl == IDC\_BUTTON\_C)  {  setBtnColor(&dc, rect, bb);  } | 솔루션 탐색기에서 MFCApplication6.rc 클릭  Dialog 폴더에 IDD\_MFCAPPLICATION6\_DIALOG 클릭  계산기 버튼을 오른쪽 마우스 클릭해 설정에서 OwnerItem을 True로 변경한 후  원하는 색으로 버튼을 변경하는 코드를 OnDrawItem에 입력합니다. |