| 시스템 프로그래밍 어셈블러 만들기 | | | |
|---|---|---|---|
| 담당교수님 | 이형봉 | 작성자 | 20141640 이석원 |
| 날짜 | 2017.12.25 | | |

| 구성 파일의 역할 | |
|---|---|
| asembler.c | main함수가 있습니다. 함수 path1과 path2를 호출합니다. |
| asembler.h | Symbol table과 Optable 구조체 정의가 있습니다. |
| path1.c | path1  함수정의가 있습니다. path1은 readline함수를 호출합니다. |
| path1.h | path1 함수 선언이 있습니다. |
| path2.c | path2  함수정의가 있습니다. path2는 readline.c의 readline함수와 charAnalisys.c의 HexToInt, numto, opcodeto, operandto 함수를 호출합니다. |
| path2.h | path2 함수 선언이 있습니다. |
| readline.c | readline 함수 정의가 있습니다. readline함수는 asemfile에서 문자열을 한 줄 씩 읽어서 각각 열을 기준으로 1열은 Label, 2열은 Opcode, 3열은 Operand 에 저장합니다. 열이 공백일경우 '\t'을 삽입합니다.<br>한줄에서 공백을 제외한 열의 개수를 리턴합니다. |
| readline.h | readline 함수 선언이 있습니다. |
| charAnalisys.c | 문자열을 문자 단위로 분석하는 함수들이 있습니다. path2에서만 사용합니다. |
| asemfile | SIC명령어로 작성된 어셈블러 코드 입니다. 행번호와 주석을 제외한 설명 부분은 제외 했습니다. |
| asembler | 실행파일입니다. |
| ObjectProgram | 목적 코드가 저장되는 파일입니다. |

**asemfile**

```
unix:AsemProject ysw1014$ cat asemfile
COPY    START   1000
FIRST   STL     RETADR
CLOOP   JSUB    RDREC
        LDA     LENGTH
        COMP    ZERO
        JEQ     ENDFIL
        JSUB    WRREC
        J       CLOOP
ENDFIL  LDA     EOF
        STA     BUFFER
        LDA     THREE
        STA     LENGTH
        JSUB    WRREC
        LDL     RETADR
        RSUB
EOF     BYTE    C'EOF'
THREE   WORD    3
ZERO    WORD    0
RETADR  RESW    1
LENGTH  RESW    1
BUFFER  RESB    4096
.
.       SUBROUTINE TO READ RECORD INTO BUFFER
.
RDREC   LDX     ZERO
        LDA     ZERO
RLOOP   TD      INPUT
        JEQ     RLOOP
        RD      INPUT
        COMP    ZERO
        JEQ     EXIT
        STCH    BUFFER,X
        TIX     MAXLEN
        JLT     RLOOP
EXIT    STX     LENGTH
        RSUB
INPUT   BYTE    X'F1'
MAXLEN  WORD    4096
.
.       SUBROUTINE TO WRITE RECORD FROM BUFFER
.
WRREC   LDX     ZERO
WLOOP   TD      OUTPUT
        JEQ     WLOOP
        LDCH    BUFFER,X
        WD      OUTPUT
        TIX     LENGTH
        JLT     WLOOP
        RSUB
OUTPUT  BYTE    X'05'
        END     FIRST
unix:AsemProject ysw1014$
```

**readline.c**

```
 6 int readline(FILE* line)
 7 {
 8         int i , j, k;
 9
10         char * colum[4];
11         char string[50];
12
13
14         if(fgets(string, 50, line) != NULL)
15         {
```

한 줄에 읽는 단어수는 최대 49자입니다.

```
16                 if(string[0] != '.' && string[0] != '\n')
17                 {
18                         for(i = 1, colum[0] = strtok(string," \t\n"); colum[i-1] != NULL; i++)
19                         {
20                                 colum[i] = strtok(NULL," \t\n");
21                         }
22
```

16행에서 맨처음 온점으로 시작하는 주석과 불필요한 개행 문자를 걸러줍니다.
18행에서 21행 까지 공백, 개행, 탭문자를 구분하여 문자열을 잘라서 colum변수에 저장합니다. colum을 4칸 배열로 설정한 이유는 문자열 string을 더이상 자를 수 없을 때 NULL값을 저장하기 위해서 입니다.

```
23                         if(i == 4)
24                         {
25                                 strcpy(Label,colum[0]);
26                                 strcpy(Opcode,colum[1]);
27                                 strcpy(Operand,colum[2]);
28                         }
29
30                         else if(i == 3)
31                         {
32                                 strcpy(Label, "\t");
33                                 strcpy(Opcode, colum[0]);
34                                 strcpy(Operand, colum[1]);
35                         }
36                         else if(i == 2)
37                         {
38                                 strcpy(Label, "\t");
39                                 strcpy(Opcode, colum[0]);
40                                 strcpy(Operand, "\t");
41                         }
```

열의 개수에 맞춰서 Label, Opcode, Operand변수에 저장합니다.

```
43                  else
44                  {
45                          i = 1;
46                  }
47          }
48          printf("read %s %s %s\n", Label, Opcode, Operand);
49          return i-1;
50  }
```

주석문자나 개행문자로 시작하는 줄은 아무것도 하지 않습니다. 28행은 디버깅을 위한 코드고 49행은 열의 개수를 리턴합니다.

**path1.c**

**path1함수는 기본적으로 교재의 알고리즘을 적용 했습니다.**

```
 1 # include <stdio.h>
 2 # include <stdlib.h>
 3 # include <string.h>
 4 # include "readline.h"
 5 # include "asembler.h"
 6 char Label[10], Opcode[10], Operand[20];
 7 int programLength;
 8
 9 struct symboltable SYMTAB[20];
10 struct operationtable OPTAB[42] = {{"ADD", 0x18},{"ADDF", 0x58},{"AND", 0x40},{"COMP", 0x28},
11                                   {"COMPF", 0x88},{"DIV", 0x24},{"DIVF", 0x64},{"J",0x3C},
12                                   {"JEQ",0x30},{"JGT",0x34},{"JLT", 0x34},{"JSUB",0x48},
13                                   {"LDA",0x00},{"LDB",0x68},{"LDCH",0x50},{"LDF",0x70},
14                                   {"LDL",0x08},{"LDS",0x6C},{"LDT",0x74},{"LDX",0x04},
15                                   {"LPS",0xD0},{"MUL",0x20},{"MULF",0x60},{"OR",0x44},
16                                   {"RD",0xD8},{"RSUB",0x4C},{"SSK",0xEC},{"STA",0x0C},
17                                   {"STB",0x78},{"STCH",0x54},{"STF",0x80},{"STI",0xD4},
18                                   {"STL",0x14},{"STS",0x7C},{"STSW",0xE8},{"STT",0x84},
19                                   {"STX",0x10},{"SUB",0x1C},{"SUBF",0x5C},{"TD",0xE0},
20                                   {"TIX",0x2C},{"WD",0xDC}};
```

7행에 프로그램 전체 크기를 전역 변수로 선언 합니다.
9,10 행에 SYMTAB과 OPTAB을 선언합니다.

```
21 void path1()
22 {
23         FILE* source = fopen("asemfile","r");
24         if(source == NULL)
25         {
26                 fprintf(stderr,"file open error \n");
27                 exit(1);
28         }
29         char* pEnd;
30         int NumofColum;
31         int StartingAdres;
32         int LOCCTR;
33         int i, k;
34         for(k = 0; k < 20; k++)
35         {
36                 SYMTAB[k].LOCCTR = -1;
37         }
38         for(k = 0; k < 42; k++)
39         {
40                 printf("OPTAB[%d] : %s %02X\n",k,OPTAB[k].Operator, OPTAB[k].code);
41         }
42
```

34행은 SYMTAB의 주소지의 초기 값을 모두 -1로 세팅합니다.
38행~41행은 디버깅 코드입니다.

```
43         NumofColum = readline(source);
44
45         while(NumofColum == 0)
46         {
47                 NumofColum = readline(source);
48         }
```

asemfile에서 한줄을 읽을 때마다 readline함수를 호출합니다.
45~48행은 만약 읽어온 줄이 주석문일때 다시 readline을 호출하는 문장입니다. path1, path2어디서든 한 줄을 읽을때마다 43~48코드는 세트로 작성합니다.

```
49         if(!strcmp(Opcode, "START"))
50         {
51                 StartingAdres = strtol(Operand, &pEnd, 16);
52                 LOCCTR = StartingAdres;
53
54                 NumofColum = readline(source);
55                 while(NumofColum == 0)
56                 {
57                         NumofColum = readline(source);
58                 }
59         }
```

맨처음 START문을 읽었을때, 시작주소와 주소계수기를 설정합니다.

```
60          else
61          {
62                  StartingAdres = 0;
63                  LOCCTR = 0;
64          }
```

Start문이 없을때 0번지에서 시작합니다.

```
66          while(strcmp(Opcode,"END"))
67          {
68                  if(NumofColum != 0)
69                  {
70                          if(Label[0] != '\t')
71                          {
72                                  for(k = 0; k < 20; k++)
73                                  {
74                                          if(SYMTAB[k].LOCCTR == -1)
75                                          {
76                                                  strcpy(SYMTAB[k].LABEL, Label);
77                                                  SYMTAB[k].LOCCTR = LOCCTR;
78                                                  break;
79                                          }
80                                          else if(!strcmp(SYMTAB[k].LABEL, Label))
81                                          {
82                                                  fprintf(stderr,"duplicate symbol");
83                                                  exit(1);
84                                          }
85                                  }
86                          }
```

70~86행은 Label열이 공백이 아닐경우 SYMTAB에 Label 이름과 위치 주소를 저장합니다.

```
88                          //search OPTAB for OPCODE
89                          int searchopflag = 0;
90
91                          for(k = 0; k < 42; k++)
92                          {
93
94                                  if(!strcmp(Opcode, OPTAB[k].Operator))
95                                  {
96
97                                          searchopflag = 1;
98                                          break;
99                                  }
100                         }
```

Opcode가 현재 OPTAB에 저장되어 있는 정당한 문자열인지 검사합니다.

```
101                         if(searchopflag)
102                         {
103                                 printf("present : %s %s LOCCTR : %X\n",Opcode,OPTAB[k].Operator, LOCCTR);
104                                 LOCCTR += 3;
105                         }
106                         else if(!strcmp(Opcode, "WORD"))
107                         {
108                                 printf("WORD present : %s LOCCTR : %X\n",Opcode,LOCCTR);
109                                 LOCCTR += 3;
110                         }
111                         else if(!strcmp(Opcode, "RESW"))
112                         {
113                                 printf("RESW %d LOCCTR : %X\n",atoi(Operand),LOCCTR);
114                                 LOCCTR += (3 * atoi(Operand));
115                         }
116
117                         else if(!strcmp(Opcode, "RESB"))
118                         {
119                                 int oper = atoi(Operand);
120                                 LOCCTR += oper;
121                                 printf("RESB %d LOCCTR : %X\n",oper, LOCCTR);
122                         }
123                         else if(!strcmp(Opcode, "BYTE"))
124                         {
125                                 int length;
126                                 if(Operand[0] == 'C')
127                                 {
128                                         length = strlen(Operand)-3;
129                                 }
130                                 else if(Operand[0] == 'X')
131                                 {
132                                         length = (strlen(Operand)-3)/2;
133                                 }
134                                 LOCCTR += length;
135                         }
136                         else
137                         {
138                                 fprintf(stderr,"invalid operation code");
139                                 exit(1);
140                         }
```

Opcode가 OPTAB에 있을때 아니면 WORD RESW RESB BYTE인 경우를 모두 확인하여 상황에 맞게 위치 계수기 값을 변경합니다. 132행에서 Operand에 “X’.....’”이 있을때 홀수인 경우를 적용하지 않았습니다.

```
143                 NumofColum = readline(source);
144                 while(NumofColum == 0)
145                 {
146                         NumofColum = readline(source);
147                 }
148         }
149         programLength = LOCCTR - StartingAdres;
150
151         for(k = 0; k < 20; k++)
152         {
153                 printf("Talbe %d \nLabel : %s location : %X\n", k, SYMTAB[k].LABEL,SYMTAB[k].LOCCTR);
154         }
155         printf("length : %X\n", programLength);
156         if(fclose(source) != 0)
157         {
158
159                 fprintf(stderr,"file close error \n");
160                 exit(1);
161         }
162 }
```

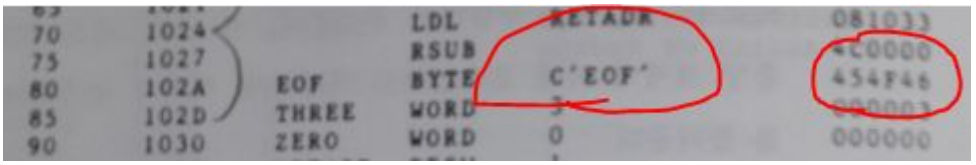
149행에서 프로그램 전체 크기 계산하고, 151~155행은 SYMTAB과 전체 크기를 디버깅 하기 위한 문장입니다.

**charAnalisys.c**

```
 8 int HexToInt(char buf[], int strsize)
 9 {
10         int i;
11         int j;
12
13         char* endptr;   //strtolÇÔ¼ö¸¦ »ç¿ëÇÏ±âÀ§ÇÑ °¯¼öÀÔ´Ï´Ù.
14         char tohex[10];   //¼ýÀÚ´Ü¾î¸¦ ÀúÀåÇÏ±âÀ§ÇÑ ¹è¿ÀÔ´Ï´Ù.
15
16         int hex = 0;  //´Ü¾î ÇÏ³ª´ç ³ª¿À´Â ¼ö¸¦ ÀúÀåÈü´Ï´Ù.
17         int asc = 0;
18         int dec = 0;
19
20         if(buf[0] == 'X' &&buf[1] == 39 && buf[strsize-1] == 39)
   © '·Î ³¡³ª¾ß ÇÕ´Ï´Ù.
21         {
22                 printf("%s \n", buf);   //µð¹ö±ëÀ» À§ÇÑ ÄÚµåÀÔ´Ï´Ù.
23                 for(i = 2; i < strsize-1; i++)  //Ã³À½¿À´Â '°ÎÅÍ ¼ÃÀÛÇÏ
24                 {
25                         tohex[i-2] = buf[i];   //»õ·Î¿î ¹è¿¿¡ ÀúÀåÇÕ´Ï
26                 }
27                 hex = (int)strtol(tohex, &endptr, 16);    //16Á¤¼ö °¯È¯
28
29                 printf("%d \n", hex);   //µð¹ö±ëÀ» À§ÇÑ ÄÚµåÀÔ´Ï´Ù.
30                 return hex;
31         }
32         else if(buf[0] == 'C' && buf[1] == 39 && buf[strsize-1] == 39)
33         {
34                 printf("%s \n", buf);   //µð¹ö±ëÀ» À§ÇÑ ÄÚµåÀÔ´Ï´Ù.
35                 for(i = 2; i < strsize-1; i++)
36                 {
37                         asc += buf[i];          //¾Æ½°ÅºÄÚµå°¡ ÀÚÃ¤·Î ´
38                         printf("%x----%d \n", asc, asc);        //µð¹ö±
39                         asc = asc << 8; //ÀÚ¸´¼ö¸¦ ÀÌµ¿½ÃÅµ´Ï´Ù.
40                 }
41                 asc = asc >> 8; //¸¶Áö¸·¿¡ °ÒÇÊ¿äÇÑ ÀÚ¸´¼ö ÀÌµ¿À» ¿ø»óÅ
42                 return asc;
43         }
44         else
45         {
46                 dec = atoi(buf);
47                 return dec;
48         }
49         return 0;
50 }
```

HexToInt함수는 지난 프로젝트에서 작성했던 함수를 조금 수정했습니다. Operand에서 X,C혹은 10진수가 있을때 문자열 Operand와 Operand의 문자열 크기를 인자로 하여 각각의 경우를 숫자로 변환하는 함수입니다.
16~18행은 각 경우에 저장되는 변수들입니다.
20~31는 Operand가 X로 시작할때 문자열을 16진수로 변환 하고 30행에서 리턴합니다.
32~42은 C로 시작할때 문자열을 아스키코드값으로 변환하여 42행에서 리턴합니다.
44행에서 48행은 문자열을 10진수로 변환하여 리턴합니다.

path2에서 Object코드를 조합하는 경우는 2가지 입니다.




1. 6자의 Object코드에 6자 모두 Operand값을 넣는다.
2. 2자의 Opcode와 4자의 Operand를 조합한다.

```
51 void numto(double number, char * ObjectCode)
52 {
53         char temp[7] = {0,0,0,0,0,0,'\0'};
54         int i , k;
55         double tm;
56         for(i = 5; i >= 0; i--)
57         {
58                 for(k = i, tm = 1; k > 0; k--)
59                 {
60                         tm = 16*tm;
61                 }
62                 temp[5-i] = (char)(number / tm);
63                 number = fmod(number,tm);
64                 if(temp[5-i] < 10)
65                 {
66                         temp[5-i] += 48;
67                 }
68                 else
69                 {
70                         temp[5-i] += 55;
71                 }
72         }
73         for(i = 0; i < 7; i++)
74         {
75                 ObjectCode[i] = temp[i];
76         }
77 }
```

numto 함수는 1번 경우를 해결 하기위한 함수 입니다. double형 number에 변환 하고자 하는 수를 받아서 ObjectCode 변수에 변환 하여 저장합니다.
58~63행은 10진수를 16진수로 변환하여 각 자리수마다 차례로 temp에 저장합니다. double형 number와 fmod를 사용하는 이유는 int형으로는 6자리 16진수 계산할 크기가 안되는지 계속 trap abort 실행 오류나 segmentation fault 실행 오류가 생깁니다.
64~71행은 각 변환 숫자를 아스키 코드 문자로 변경합니다.

```
 79 void opcodeto(int symint, char * Symvalue)
 80 {
 81         char temp[3] = {0,0,'\0'};
 82         int i , k;
 83         int tm;
 84         for(i = 1; i >= 0; i--)
 85         {
 86                 for(k = i, tm = 1; k > 0; k--)
 87                 {
 88                         tm = 16*tm;
 89                 }
 90                 temp[1-i] = (char)(symint / tm);
 91                 symint = symint%tm;
 92                 if(temp[1-i] < 10)
 93                 {
 94                         temp[1-i] += 48;
 95                 }
 96                 else
 97                 {
 98                         temp[1-i] += 55;
 99                 }
100         }
101         for(i = 0; i < 3; i++)
102         {
103                 Symvalue[i] = temp[i];
104         }
105 }
```

기본적으로 numto 함수와 같은 진행을 합니다. 91행에서 %연산을 사용하는 점과 전체 배열 크기가 다릅니다.

```
106 void operandto(int symint, char * Symvalue)
107 {
108         char temp[5] = {0,0,0,0,'\0'};
109         int i , k;
110         int tm;
111         for(i = 3; i >= 0; i--)
112         {
113                 for(k = i, tm = 1; k > 0; k--)
114                 {
115                         tm = 16*tm;
116                 }
117                 temp[3-i] = (char)(symint / tm);
118                 symint = symint%tm;
119                 if(temp[3-i] < 10)
120                 {
121                         temp[3-i] += 48;
122                 }
123                 else
124                 {
125                         temp[3-i] += 55;
126
127                 }
128         }
129         if(strchr(Operand,',')!=NULL)
130         {
131                 temp[0] += 8;
132         }
133         for(i = 0; i < 5; i++)
134         {
135                 Symvalue[i] = temp[i];
136         }
137 }
```

마찬가지로 같은 진행을 하는데, 차이점은 129~132행에서 X비트 사용에대한 대처가 있습니다.

**path2.c**

path2함수도 교재의 알고리즘을 그대로 적용했으나 주소계수기의 부재와 RESW, RESB에 대한 처리 등이 미흡한점을 확인 하지 못하여 스파게티 프로그램처럼 작성 되었습니다.
ObjectProgram을 작성하는데 기본적으로 listingLine은 char형 배열 형식입니다. 한 자씩 하나의 배열 공간에 데이터가 저장됩니다.

```
39          if(!strcmp(Opcode,"START"))
40          {
41                  strcat(listingLine, Label);
42                  numberTemp = strtol(Operand,&pEnd,16);
43                  numto(numberTemp, ObjectCode);
44                  strcat(listingLine, ObjectCode);
45                  numto(programLength, ObjectCode);
46                  strcat(listingLine, ObjectCode);
47                  printf("linstingLine %s\n", listingLine);
48
49
50                  startingadres = numberTemp;
51                  LocCtr = numberTemp;
52                  LineStartLocCtr = LocCtr;
53                  numofcolum = readline(source);
54                  while(numofcolum == 0)
55                  {
56                          numofcolum = readline(source);
57                  }
58          }
```

첫 줄을 읽은 후 헤더레코드를 작성하는 부분입니다.
41행에서 프로그램의 이름이 listingLine에 추가되고
43,44행에서 프로그램 시작주소 추가
45,46행에서 프로그램 전체 크기가 추가됩니다.
50,51행은 시작 주소와 위치계수기를 설정하고
52행은 텍스트 레코드에 첫 단어로 저장되는 텍스트레코드 시작주소가 저장됩니다.

```
59          fprintf(object,"H%s\n",listingLine);
60          strcpy(listingLine,"");
61          printf("reset listingLine %s\n",listingLine);
```

59행에서 헤드 레코드를 ObjectProgram에 기록합니다.
60행에서 listingLine을 비워줍니다.

```
62          while(strcmp(Opcode,"END"))
63          {
64                  int i;
65                  int Opfound = 0;
66                  int Symfound = 0;
67                  char Opvalue[3];
68                  int Opint;
69                  char Symvalue[5];
70                  int Symint;
71                  strcpy(ObjectCode,"");
72                  strcpy(Opvalue,"");
73                  strcpy(Symvalue,"");
74                  for(i = 0; i < 42; i++)
75                  {
76                          if(!strcmp(OPTAB[i].Operator, Opcode))
77                          {
78                                  Opfound = 1;
79                                  Opint = OPTAB[i].code;
80                                  break;
81                          }
82
83                  }
```

74~83행에서 Opcode를 OPTAB에서 검색하여 16진수 코드값을 Opint에 저장합니다.

```
84                  if(Opfound == 1)
85                  {
86                          if(strcmp(Operand, "\t") != 0)
87                          {
88                                  printf("befor loop\n");
89                                  char * cutX;
90                                  char Opercopy[20];
91                                  strcpy(Opercopy, Operand);
92                                  if(strchr(Opercopy,',')!=NULL)
93                                  {
94                                          cutX = strtok(Opercopy,",");
95
96                                          for(i = 0; i < 20; i++)
97                                          {
98                                                  if(!strcmp(SYMTAB[i].LABEL, cutX))
99                                                  {
100                                                         Symfound = 1;
101                                                         Symint = SYMTAB[i].LOCCTR;
102                                                         break;
103                                                 }
104                                         }
105                                 }
```

88~105행에서 Operand의 X비트 사용 여부를 일단 제외하고 Operand값을 SYMTAB에서 검색합니다. X비트 사용여부는 Operandto함수 에서 따로 확인 합니다.

```
106                             else
107                             {
108                                     for(i = 0; i < 20; i++)
109                                     {
110                                             if(!strcmp(SYMTAB[i].LABEL, Operand))
111                                             {
112                                                     Symfound = 1;
113                                                     Symint = SYMTAB[i].LOCCTR;
114                                                     break;
115                                             }
116                                     }
117                             }
```

X비트를 사용하지 않을때 검색 하는 문장입니다.

```
118                             if(Symfound == 1)
119                             {
120                                     operandto(Symint, Symvalue);
121                                     printf("sym value %s\n", Symvalue);
122                             }
123                             else
124                             {
125                                     fprintf(stderr,"Operand error \n");
126                                     exit(1);
127                             }
128                     }
```

SYMTAB에서 Operand를 찾았을때 Label의 주소값을 문자열로 변환하여 Symvalue에 저장합니다.

```
130                     else
131                     {
132                             operandto(0,Symvalue);
133                             printf("operand is empty\n");
134                     }
135                     opcodeto(Opint, Opvalue);
136                     strcat(ObjectCode,Opvalue);
137                     strcat(ObjectCode,Symvalue);
138                     printf("this is Objectcode %s\n", ObjectCode);
139             }
```

130~134는 Operand열이 비었을때 Symcalue에 0을 저장하고 135~137은 Opcode와 Operand를 조합하여 ObjectCode를 구성합니다.

```
140                 else if(!strcmp(Opcode,"BYTE"))
141                 {
142                         int convertnum = 0;
143                         convertnum = HexToInt(Operand,strlen(Operand));
144                         printf("convert number is %x \n", convertnum);
145                         if(convertnum > 255)
146                         {
147                                 numto(convertnum, ObjectCode);
148                                 printf("over 255 is it X,C %s\n", ObjectCode);
149                         }
150                         else
151                         {
152                                 opcodeto(convertnum,Opvalue);
153                                 strcpy(ObjectCode, Opvalue);
154                                 printf("under 255 is it X,c %s\n", ObjectCode);
155                         }
156                 }
157                 else if(!strcmp(Opcode,"WORD"))
158                 {
159                         int temp;
160                         temp = atoi(Operand);
161                         numto(temp,ObjectCode);
162                 }
163                 else if(!strcmp(Opcode,"RESB"))
164                 {
165                         LocCtr = LocCtr + atoi(Operand);
166                 }
167                 else if(!strcmp(Opcode, "RESW"))
168                 {
169                         LocCtr = LocCtr + (atoi(Operand) * 3);
170                 }
```

140~170은 Opcode열에 BYTE, WORD는 각 Operand를 처리하고 RESW와 RESB는 위치 계수기만 계산합니다.
BYTE계산 할때 Operand가 255보다 큰지 작은지 검사하여 2자짜리 ObjectCode혹은 6자 짜리 ObjectCOde를 구성합니다.

```
171                 if(strlen(listingLine) > 55)
172                 {
173                         printf("this is 60 over listring line **********************\n");
174                         int textLength = strlen(listingLine)/2;
175                         printf("Length of listingLine %02X\n", textLength);
176                         fprintf(object,"T%06X%02X%s\n",LineStartLocCtr,textLength,listingLine);
177                         LocCtr += textLength;
178                         LineStartLocCtr = LocCtr;
179                         strcpy(listingLine,"");
180                 }
```

listingLine의 길이를 검사하여 ObjectCode가 더이상 삽입 할 수 없으면 listingLine을 ObjectProgram에 기록합니다.
174행 텍스트 레코드에서 두번째 문자로 오는 텍스트레코드 길이는 listringLine길이를 통해 구합니다.

## 프로그램 컴파일과 실행

```
unix:AsemProject ysw1014$ ls -al
total 186
drwx------+ 1 ysw1014  A\Domain Users  16384 Dec 24 13:00 .
drwx------+ 1 ysw1014  A\Domain Users  16384 Dec  2 10:26 ..
-rwx------+ 1 ysw1014  A\Domain Users   4096 Dec 21 18:15 .readlinetest.c.swp
-rwx------+ 1 ysw1014  A\Domain Users    299 Dec 24 11:22 ObjectProgram
-rwx------+ 1 ysw1014  A\Domain Users  18056 Dec 24 11:21 asembler
-rwx------+ 1 ysw1014  A\Domain Users    139 Dec 22 11:57 asembler.c
-rwx------+ 1 ysw1014  A\Domain Users    158 Dec 22 17:47 asembler.h
-rwx------+ 1 ysw1014  A\Domain Users    724 Dec 22 11:57 asembler.o
-rwx------+ 1 ysw1014  A\Domain Users    663 Dec 22 20:19 asemfile
-rwx------+ 1 ysw1014  A\Domain Users   2679 Dec 23 18:29 charAnalisys.c
-rwx------+ 1 ysw1014  A\Domain Users    292 Dec 23 16:39 charAnalisys.h
-rwx------+ 1 ysw1014  A\Domain Users   3084 Dec 23 18:29 charAnalisys.o
-rwx------+ 1 ysw1014  A\Domain Users   3566 Dec 24 11:21 path1.c
-rwx------+ 1 ysw1014  A\Domain Users     56 Dec 22 17:46 path1.h
-rwx------+ 1 ysw1014  A\Domain Users   4352 Dec 24 11:21 path1.o
-rwx------+ 1 ysw1014  A\Domain Users   4262 Dec 23 22:49 path2.c
-rwx------+ 1 ysw1014  A\Domain Users     56 Dec 22 11:53 path2.h
-rwx------+ 1 ysw1014  A\Domain Users   5584 Dec 23 22:49 path2.o
-rwx------+ 1 ysw1014  A\Domain Users    897 Dec 22 20:17 readline.c
-rwx------+ 1 ysw1014  A\Domain Users    159 Dec 21 17:13 readline.h
-rwx------+ 1 ysw1014  A\Domain Users   1856 Dec 22 20:17 readline.o
unix:AsemProject ysw1014$ vi path2.c
unix:AsemProject ysw1014$ gcc -c asembler.c
unix:AsemProject ysw1014$ gcc -c charAnalisys.c
unix:AsemProject ysw1014$ gcc -c path1.c
unix:AsemProject ysw1014$ gcc -c path2.c
unix:AsemProject ysw1014$ gcc -c readline.c
unix:AsemProject ysw1014$ gcc -o asembler asembler.o charAnalisys.o path1.o path2.o readline.o
unix:AsemProject ysw1014$ ./asembler
```

```
unix:AsemProject ysw1014$ cat ObjectProgram
HCOPY00100000107A
T0010001E14103348203900103628103030101548206 13C100300102A0C103900102D
T00101E1E0C10364820610810334C0000454F46000003000000041030001030E0205D
T0020421C30203FD8205D2810303020575490392C205E34203F1010364C0000F1
T00205E1C001000041030E02079302064509039DC20792C10363420644C000005
E001000
unix:AsemProject ysw1014$
```