Warren Fernandes TE COMPS B (8940) BATCH B

Experiment No 2

Aim: Write a program to implement Two Pass Assembler

Learning Objective: Translating mnemonic operation codes to their machine language equivalents. Assigning machine addresses to symbolic labels used by the programmer. Lastly to convert assembly language to binary.

Algorithm:

Pass 1:

1. Start
2.Intialize location counter to zero

3.Read opcode field of next instruction.

4.search opcode in pseudo opcode Table(POT)

5.If opcode is found in POT

5.1 If it is ‘DS’ or ‘DC’

Adjust location counter to proper alignment.

Assign length of data field to ‘L’

Go to step 9

5.2 If it is ‘EQU’

Evaluate operand field

Assign values to symbol in label field

Go to step 3

5.3 If it is ‘USING’ or ‘DROP’ Go to step 3
5.4 If it is ‘END’

Assign value to symbol in label field

Go to step 3

6. Search opcode in Machine Opcode Table.

7. Assign its length to ‘L’.
8. Process any literals and enter them into literal Table.
9.If symbol is there in the label field

Assign current value of Location Counter to symbol

10. Location Counter= Location Counter +L.

11.Go to step 3.
12. Stop.

Pass2:

1.Start
2. Intialize location counter to zero.
3. Read opcode field of next instruction.
4. Search opcode in pseudo opcode table.

5. If opcode is found in pseudo opcode Table

5.1 If it is ‘DS’ or ‘DC’

Adjust location counter to proper alignment.

If it is ‘DC’ opcode form constant and insert in assembled program

Assign length of data field to ‘L’

Go to step 6.4

5.2 If it is ‘EQU’ or ‘START’ ignore it. Go to step 3

5.3 If it is ‘USING’

Evaluate operand and enter base reg no. and value into base table

Go to step 3

5.4 If it is ‘DROP’

Indicate base reg no . available in base table . Go to step 3

5.5 If it is ‘END’

Generate literals for entries in Literal Table

Go to step 12

6 Search opcode in MOT

7. Get opcode byte and format code

8. Assign its length to ‘L’.
9. Check type of instruction.

10.If it is type ‘RR’ type

10.1 Evaluate both register expressions and insert into second byte.
10.2 Assemble instruction

10.3 Location Counter= Location Counter +L.

10. 4.Go to step 3.

11. If it is ‘RX’ type

11.1 Evaluate register and index expressions and insert into second byte.

11.2 Calculate effective address of operand.
11.3 Determine appropriate displacement and base register
11.4 Put base and displacement into bytes 3 and 4

11.5 Location Counter= Location Counter +L.

11.6 Go to step 11.2 13 Stop.

Implementation Details

1. Read Assembly language input file

2. Display output of Pass1 as the output file with Op-code Table, Symbol Table

3. Display output of pass2 as the Op-code Table, Symbol Table , Copy file

Test Cases:

1 Input symbol which is not defined

2 Input Opcode which is not entered in MOT

Conclusion: Two pass assembler has been implemented

REFERENCE:

PASS 1: https://youtu.be/esDnuGD6kb0

PASS 2: https://forgetcode.com/c/104-pass-two-of-a-two-pass-assembler

PASS 1 CODE

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h> void
main() {
   char opcode[10], operand[10], label[10], code[10], mnemonic[10];     int
locctr, start, length;

   FILE *fp1, *fp2, *fp3, *fp4;

   fp1 = fopen("input.txt", "r");
fp2 = fopen("optab.txt", "r");     fp3
= fopen("symtbl.txt", "w");
   fp4 = fopen("output.txt", "w");

   fscanf(fp1, "%s\t%s\t%s", label, opcode, operand);

   if(strcmp(opcode, "START")==0) {
start = atoi(operand);
      locctr = start;
      fprintf(fp4, "\t%s\t%s\t%s\n", label, opcode, operand);
      fscanf(fp1, "%s\t%s\t%s", label, opcode, operand);
```

```
   }     else {
locctr = 0;
   }

   while(strcmp(opcode, "END")!=0) {
fprintf(fp4, "%d\t", locctr);

      if(strcmp(label, "**")!=0) {
         fprintf(fp3, "%s\t%d\n", label, locctr);
      }

      fscanf(fp2, "%s\t%s", code, mnemonic);

      while(strcmp(code, "END")!=0) {
if(strcmp(opcode, code)==0) {
            locctr+=3;
break;
         }
         fscanf(fp2, "%s\t%s", code, mnemonic);
      }

      if(strcmp(opcode, "WORD")==0) {
locctr+=3;
      }
      else if(strcmp(opcode, "RESW")==0) {
         locctr+=(3*(atoi(operand)));
      }
      else if(strcmp(opcode, "RESB")==0) {
locctr+=(atoi(operand));
      }
      else if(strcmp(opcode, "BYTE")==0) {
         ++locctr;
      }

      fprintf(fp4, "%s\t%s\t%s\t\n", label, opcode, operand);         fscanf(fp1,
"%s\t%s\t%s", label, opcode, operand);
   }

   fprintf(fp4, "%d\t%s\t%s\t%s\n", locctr, label, opcode, operand);

   length = locctr-start;

   printf("The length of the code : %d\n", length);

   fclose(fp1);
fclose(fp2);     fclose(fp3);
fclose(fp4);
}
```

INPUT

input - Notepad

File Edit Format View Help

```
**   START   2000
**   LDA FIVE
**   STA ALPHA
**   LDCH    CHARZ
**   STCH    C1
ALPHA   RESW    2
FIVE    WORD    5
CHARZ   BYTE    C'Z'
C1   RESB    1
**   END **
```

optab - Notepad

File Edit Format View Help

```
START   *
LDA 03
STA 0f
LDCH    53
STCH    57
END *
```

symtbl - Notepad

File Edit Format View Help

```
ALPHA   2012
FIVE    2018
CHARZ   2021
C1      2022
```

OUTPUT

```
The length of the code : 23

Process returned 0 (0x0)   execution time : 6.462 s
Press any key to continue.
```

output - Notepad

File Edit Format View Help

```
        **      START   2000
2000    **      LDA     FIVE
2003    **      STA     ALPHA
2006    **      LDCH    CHARZ
2009    **      STCH    C1
2012    ALPHA   RESW    2
2018    FIVE    WORD    5
2021    CHARZ   BYTE    C'Z'
2022    C1      RESB    1
2023    **      END     **
```

PASS 2 CODE

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
#include<ctype.h> main()
{
FILE  *fint,*ftab,*flen,*fsym,  *fout;
int     op1[10],txtlen,txtlen1,i,j=0,len;
char
add[5],symadd[5],op[5],start[10],temp[30],line[20],label[20],mne[10],operand[10],symtab[10],opm ne[10];

fint=fopen("input.txt","r"); flen=fopen("length.txt","r");
ftab=fopen("optab.txt","r");
fsym=fopen("symbol.txt","r");
fout=fopen("output.txt","w");
fscanf(fint,"%s%s%s%s",add,label,mne,operand);
if(strcmp(mne,"START")==0)
{
strcpy(start,operand);
fscanf(flen,"%d",&len);
}
printf("H^%s^%s^%d\nT^00%s^",label,start,len,start);
fscanf(fint,"%s%s%s%s",add,label,mne,operand);
while(strcmp(mne,"END")!=0)
{
fscanf(ftab,"%s%s",opmne,op);
while(!feof(ftab))
{
if(strcmp(mne,opmne)==0)
{
fclose(ftab); fscanf(fsym,"%s%s",symadd,symtab);
while(!feof(fsym))
{
if(strcmp(operand,symtab)==0)
{
printf("%s%s^",op,symadd);
break;
}
else fscanf(fsym,"%s%s",symadd,symtab);
}
break;
}
else fscanf(ftab,"%s%s",opmne,op);
}
if((strcmp(mne,"BYTE")==0)||(strcmp(mne,"WORD")==0))
{
if(strcmp(mne,"WORD")==0)
printf("0000%s^",operand);
else
{
len=strlen(operand);
```

```
for(i=2;i<len;i++)
{
printf("%d",operand[i]);
}
printf("^");
}
}
fscanf(fint,"%s%s%s%s",add,label,mne,operand);
ftab=fopen("optab.txt","r");
fseek(ftab,SEEK_SET,0);
}
printf("\nE^00%s",start);
fclose(fint); fclose(ftab);
fclose(fsym); fclose(flen);
fclose(fout);
getch();
}
```

INPUT

input - Notepad

File Edit Format View Help

```
-       COPY    START   1000
1000    -       LDA     ALPHA
1003    -       ADD     ONE
1006    -       SUB     TWO
1009    -       STA     BETA
1012    ALPHA   BYTE    C'KLNCE
1017    ONE     RESB    2
1019    TWO     WORD    5
1022    BETA    RESW    1
1025    -       END     -
```

symbol - Notepad

File Edit Format View Help

```
1012    ALPHA
1017    ONE
1019    TWO
1022    BETA
```

length - Notepad

File Edit Format View Help

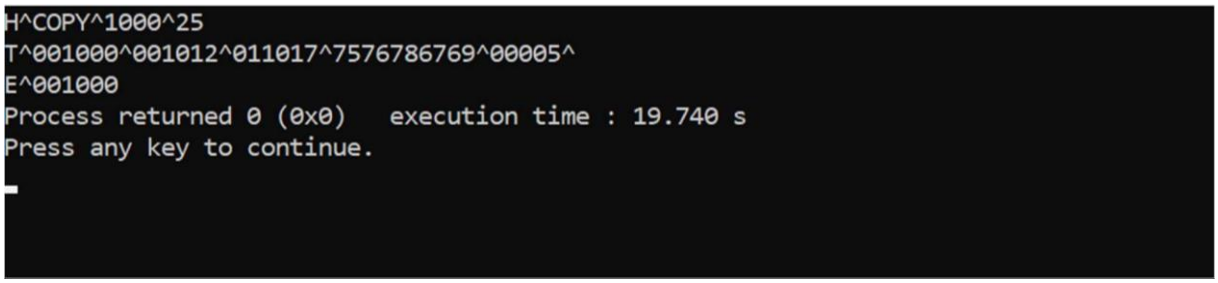
```
25
```

optab - Notepad

File Edit Format View Help

```
LDA     00
STA     23
ADD     01
SUB     05
```

OUTPUT


H^COPY^1000^25
T^001000^001012^011017^7576786769^00005^
E^001000
Process returned 0 (0x0)   execution time : 19.740 s
Press any key to continue.