System Software

```
RQST queue and install the nominated hook
                                     Main clause
 (LT,R9,VCTR_RQST_QUEUE,Z)
                                     Queue addres
 (LT,R6,QUEUE_HEAD,Z)
                                     Any items or
RQST,R6
TIL=(LT,R6,RQST_NEXT,Z)
                                     Loop for al
                                     Correct eye
RID ROST, LEAVE=YES
ECT_CLI,RQST_TYPE,EQ
                                     Examine requ
HEN (ROST@TYPE_CSVDYNEX)
@CALL PROC=INSTALL_DYNAMIC_EXIT,
      PARAM=(RQST),
      MF=(E,WA_@CALL_LIST)
HEN (RQST@TYPE_ENF_EXIT, RQST@TYPE_ENF_SRBEXIT)
@CALL PROC=INSTALL ENF LISTENER,
      PARAM=(RQST),
      MF=(E,WA_@CALL_LIST)
IF (TM,RQST_FLAG1,RQST@FLAG1_PENDING,0)
  @CALL PROC=REMOVE_REQUEST,
        PARAM=(RQST),
        MF=(E,WA_@CALL_LIST)
ELSEIF (TM, RQST_FLAG1, RQST@FLAG1_IGNORE, 0) Ignor
  @MSG 0017,
        SEV=I
        OVERLAY= (RQST NAME,
                 RQST_TYPE,
                 RQST_GROUP),
        MF=(E,WA @MSG_LIST)
ELSE
  ABEND 101
ENDIF
SEL
```

FIRST PASS ASSEMBLY





Contributers

Astha Mehrotra

180630107029

Briyansi Dabhi

180630107009

2/16

Rutvi Shah

180630107091



Professor

Mr. Chetan Chudasama

You've been the role model that taught us how to be our best no matter how daunting a challenge may seem. We shall be forever grateful for all the wonderful lessons you've shared.





About Assembler:



Things to note before you jump in

An assembler is a program that converts assembly language into machine code. It takes the basic commands and operations from assembly code and converts them into binary code that can be recognized by a specific type of processor.

4/16





Assembler HAS TWO TYPES:

Assembler Pass-1:

Define symbols and literals and remember them in symbol table and literal table respectively. o Keep track of location counter o Process pseudo-operations

Assembler Pass-2:

Generate object code by converting symbolic op-code into respective numeric op-code Generate data for literals and look for values of symbols

Implementation of PASS-1:

Working of Pass-1: Define Symbol and literal table with their addresses.

Step-1:

START 200 (here no symbol or literal is found so both table would be empty)

Step-2:

S: MOVER R1, ='3' 200 (='3' is a literal so literal table is made)

Literal Address

IMPLEMENTATION





Step-3: MOVEM R1, X 201

X is a symbol referred prior to its declaration so it is stored in symbol table with blank address field.

Symbol	Address

X _____

L1 is a label and ='2' is a literal so store them in respective tables

Symbol Address

Χ -----

L1 202

Literal Address

='3'

='2'

IMPLEMENTATION





Step-5: LTORG 203

Assign address to first literal specified by LC value, i.e., 203

Literal	Address
="3"	203
="2"	

Step-6: X DS 1 204

It is a data declaration statement i.e X is assigned data space of 1. But X is a symbol which was referred earlier in step 3 and defined in step 6. This condition is called Forward Reference Problem where variable is referred prior to its declaration and can be solved by back-patching. So now assembler will assign X the address specified by LC value of current step.

Symbol	Address
X	204
L1	202

IMPLEMENTATION





Step-7: END 205

Program finishes execution and remaining literal will get address specified by LC value of END instruction. Here is the complete symbol and literal table made by pass 1 of assembler.

Literal Address

X 204

L1 202

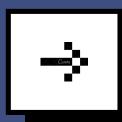
Literal Address

="3" 203

Literal Address

="2" 205

CODE





CODE.

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
#include<string.h>
void main()
 FILE *f1,*f2,*f3,*f4,*f5;
 int
lc,sa,i=0,j=0,m[10],pgmlen,len,k,len1,l=0;
 char
name[10],opnd[10],la[10],mne[10],s1[10],m
ne1[10],opnd1[10];
char lcs[10],ms[10];
char
sym[10],symaddr[10],obj1[10],obj2[10],s2[
10],q[10],s3[10];
clrscr();
f1=fopen("input.txt","r");
f2=fopen("optab.txt","r");
f3=fopen("symtab.txt","w+");
f4=fopen("symtab1.txt","w+");
f5=fopen("output.txt","w+");
fscanf(f1,"%s%s%s",la,mne,opnd);
if(strcmp(mne,"START")==0)
sa=atoi(opnd);
```

CONT.

```
strcpy(name,la);
lc=sa;
strcpy(s1,"*");
fscanf(f1,"%s%s%s",la,mne,opnd);
while(strcmp(mne,"END")!=0)
if(strcmp(la,"-")==0)
fscanf(f2,"%s%s",mne1,opnd1);
while(!feof(f2))
if(strcmp(mne1,mne)==0)
m[i]=lc+1;
fprintf(f3,"%s\t%s\n",opnd,s1);
fprintf(f5,"%s\t0000\n",opnd1);
lc=lc+3;
i=i+1;
break;
else
fscanf(f2,"%s%s",mne1,opnd1);
```

CODE





CONT.

```
else
fseek(f3,SEEK_SET,0);
fscanf(f3,"%s%s",sym,symaddr);
while(!feof(f3))
if(strcmp(sym,la)==0)
itoa(lc,lcs,10);
fprintf(f4,"%s\t%s\n",la,lcs);
itoa(m[j],ms,10);
j=j+1;
fprintf(f5,"%s\t%s\n",ms,lcs);
i=i+1;
break;
else
fscanf(f3,"%s%s",sym,symaddr);
}//f3
if(strcmp(mne,"RESW")==0)
lc=lc+3*atoi(opnd);
else if(strcmp(mne,"BYTE")==0)
```

CONT.

```
strcpy(name,la);
lc=sa;
strcpy(s1,"*");
fscanf(f1,"%s%s%s",la,mne,opnd);
while(strcmp(mne,"END")!=0)
if(strcmp(la,"-")==0)
fscanf(f2,"%s%s",mne1,opnd1);
while(!feof(f2))
if(strcmp(mne1,mne)==0)
m[i]=lc+1;
fprintf(f3,"%s\t%s\n",opnd,s1);
fprintf(f5,"%s\t0000\n",opnd1);
lc=lc+3;
i=i+1;
break;
else
fscanf(f2,"%s%s",mne1,opnd1);
```

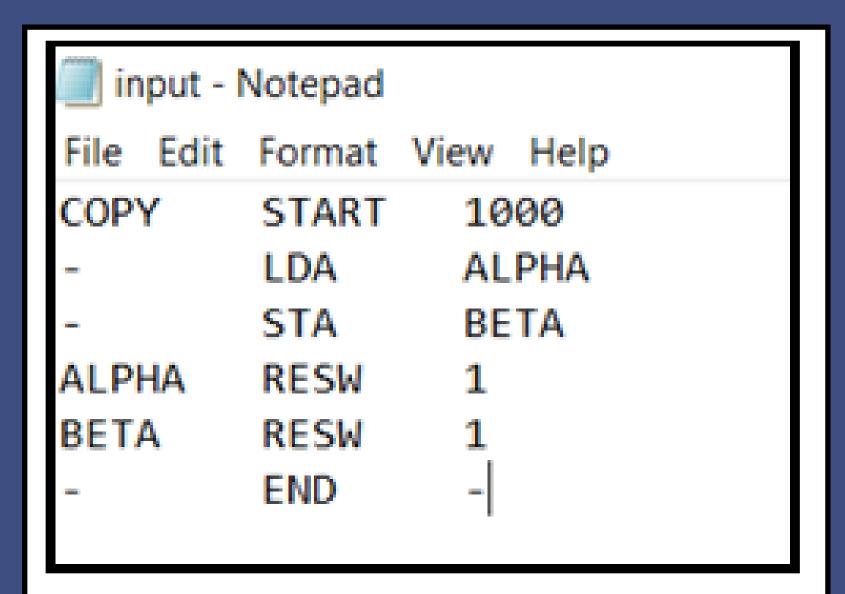
CODE



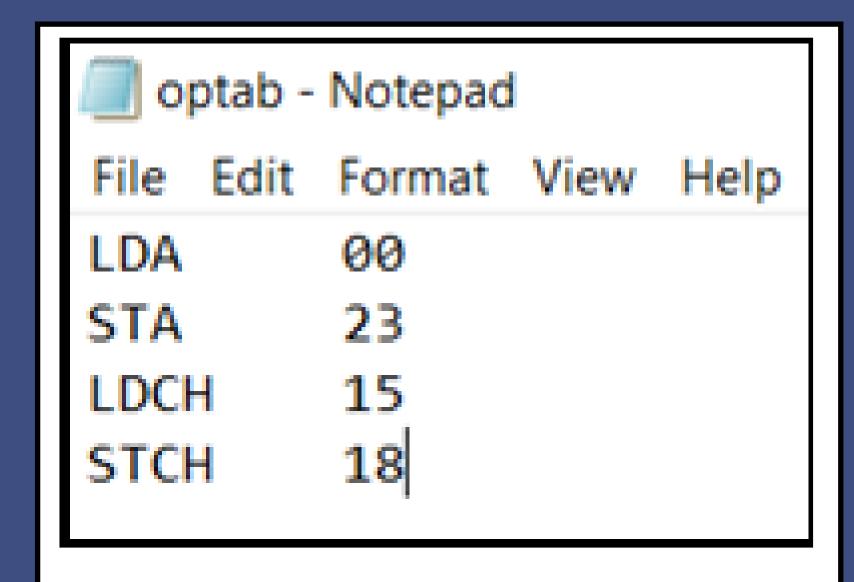


CONT. else if(strcmp(mne,"WORD")==0) strcpy(s3,"#"); lc=lc+3;fprintf(f5,"%s\t%s\n",opnd,s3); break; } // else la=fseek(f2,SEEK_SET,0); fscanf(f1,"%s%s%s",la,mne,opnd); fseek(f5,SEEK_SET,0); pgmlen=lc-sa; printf("H^%s^%d^0%x\n",name,sa,pgmlen) printf("T^"); printf("00%d^0%x",sa,pgmlen); fscanf(f5,"%s%s",obj1,obj2); while(!feof(f5)) if(strcmp(obj2,"0000")==0) printf("^%s%s",obj1,obj2); else if(strcmp(obj2,"-")==0) printf("^");

```
len1=strlen(obj1);
for(k=0;k<len1;k++)</pre>
printf("%d",obj1[k]);
else if(strcmp(obj2,"#")==0)
printf("^");
printf("%s",obj1);
while(!feof(f5))
if(strcmp(obj2,"0000")!=0)
 if(strcmp(obj2,"-")!=0)
 if(strcmp(obj2,"#")!=0)
  printf("\n");
  printf("T^%s^02^%s",obj1,obj2);
fscanf(f5,"%s%s",obj1,obj2);
printf("\nE^00%d",sa);
getch();
```

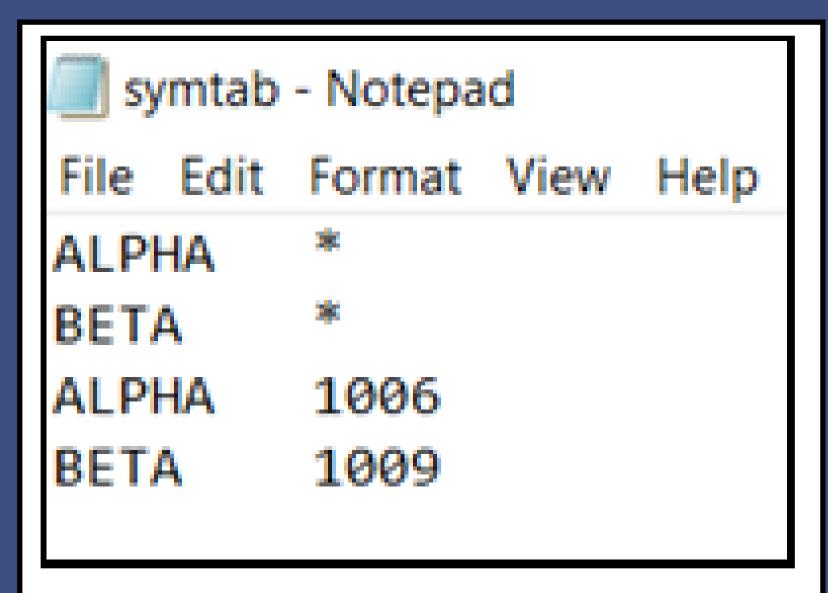


INPUT FILES: Input.txt

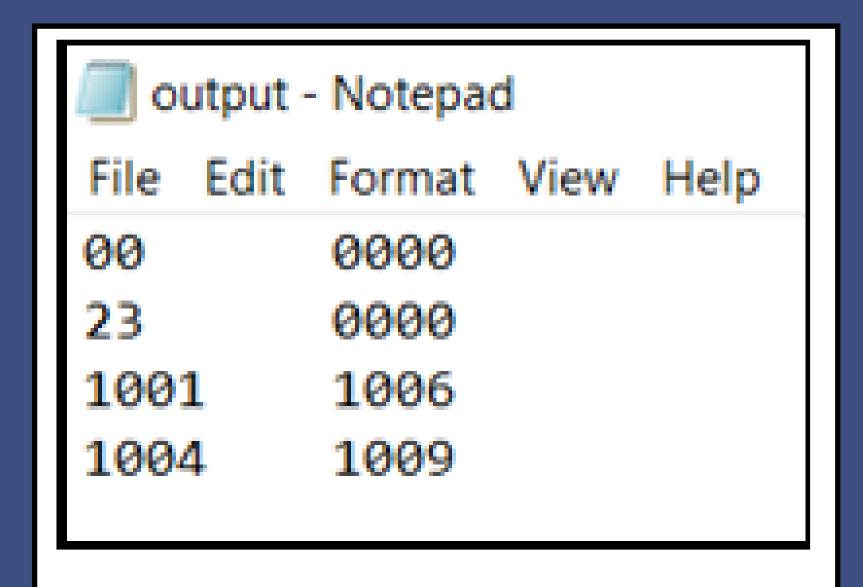


INPUT FILES: Optab.txt



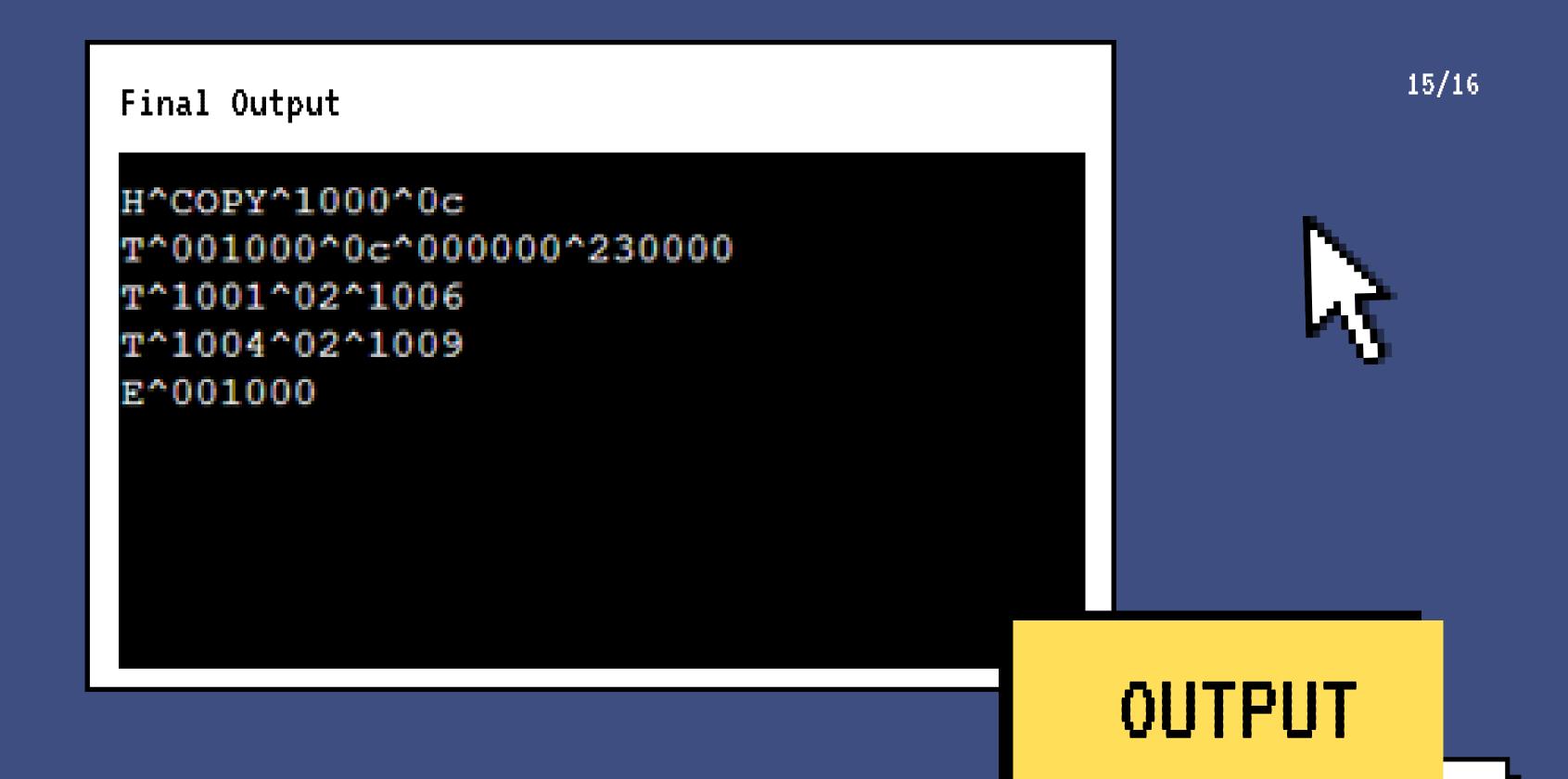


INPUT FILES: Symtab.txt



INPUT FILES: Output.txt





GRACIAS