

Name:

WAQAS ASHIQ

Roll No:

Fa11-Ciit-DDP-BCS-201

Section:

C

Submitted To:

MR.Nadeem Ghafoor Chaudhary

**simple.h**:

#include<stdio.h>

#include<stdio.h>

#define READ 10

#define WRITE 11

#define LOAD 20

#define STORE 21

#define ADD 30

#define SUBTRACT 31

#define DIVIDE 32

#define MULTIPLY 33

#define BRANCH 40

#define BRANCHNEG 41

#define BRANCHZERO 42

#define HALT 43

#define SIZE 100

**Simpletron.c**

#include"simple.h"

int main(int argc, char \*argv[])

{

int ac=0;

int ir=0;

int memory[SIZE]={0};

int ic=0;

int oc=0;

int ope=0;

int i;

loader(\*argv[1],memory);

execute(memory,&ac,&ir,&ic,&oc,&ope);

util(memory,ac,ir,ic,oc,ope);

return 0;

}

**loader.c:**

#include"simple.h"

int loader(char \*argv[1],int \*loadmemory)

{

int ins;

int i=0;

printf("\n\t\tTHE DATA TO BE READED");

FILE \*cfPtr;

if ( (cfPtr=fopen(\*argv[1],"r") ) == NULL )

{

printf("Failed to open the requested file...\n");

}

else {

cfPtr = fopen(\*argv[1],"r");

fscanf( cfPtr, "%d",&ins);

loadmemory[i]=atoi(ins);

i++;

}

while ( !feof( cfPtr ) ) {

fscanf( cfPtr, "%d",&ins);

loadmemory[i]=atoi(ins);

i++;

}

fclose ( cfPtr );

return 0;

}

**execute.c:**

#include"simple.h"

int execute(int \*memory,int \*acptr,int \*irptr,int \*icptr,int \*ocptr,int \*opeptr)

{

printf("Simpletron Begins Execution\n");

int A;

\*irptr=memory[\*icptr];

\*ocptr=\*irptr/100;

\*opeptr=\*irptr%100;

while(\*ocptr!=HALT)

{

switch(\*ocptr)

{

case READ:

printf("?");

scanf("%d",&A);

memory[\*opeptr]=A;

++(\*icptr);

break;

case WRITE:

printf("Location %0.2d: %d\n",\*opeptr,memory[\*opeptr]);

++(\*icptr);

break;

case LOAD:

\*acptr=memory[\*opeptr];

++(\*icptr);

break;

case STORE:

memory[\*opeptr]=\*acptr;

++(\*icptr);

break;

case ADD:

A=\*acptr+memory[\*opeptr];

\*acptr=A;

++(\*icptr);

break;

case SUBTRACT:

A=\*acptr-memory[\*opeptr];

\*acptr=A;

++(\*icptr);

break;

case DIVIDE:

\*acptr=\*acptr/memory[\*opeptr];

++(\*icptr);

break;

case MULTIPLY:

A=\*acptr\*memory[\*opeptr];

\*acptr=A;

++(\*icptr);

break;

case BRANCH:

\*icptr=\*opeptr;

break;

case BRANCHNEG:

if(\*acptr<0)

{

\*icptr=\*opeptr;

}

else

{

++(\*icptr);}

break;

case BRANCHZERO:

if(\*acptr==0)

{

\*icptr=\*opeptr;

}

else

{

++(\*icptr);

}

break;

case HALT:

printf("Invalid Code\n");

break;

}

\*irptr=memory[\*icptr];

\*ocptr=\*irptr/100;

\*opeptr=\*irptr%100;

}

printf("\*\*\* SIMPLE EXECUTION TERMINATED \*\*\*\n");

return 0;

}

**util.c**

#include"simple.h"

int util(int \*memory,int accumulator,int instructioncounter,int instructionregister,int operationcode,int operand)

{

printf("REGISTER\n");

printf("ACCUMULATOR=%.2d\n",accumulator);

printf("INSTRUCTION COUNTER=%.2d\n",instructioncounter);

printf("INSTRUCTION REGISTER=%.2d\n",instructionregister);

printf("OPERATION CODE=%.2d\n",operationcode);

printf("OPERAND%.2d\n",operand);

int i;

printf("\n\nMEMORY:\n");

for(i=0;i<=9;i++)

{

printf("\t%d",i);

}

for(i=0;i<SIZE;i++)

{

if(i%10==0)

{

printf("%d\n",i);

}

printf("\t\t%0.4d",memory[i]);

}

printf("\n");

return 0;

}

**makefile:**

myfile2.exe: simpletron.o loader.o execture.o libutil.so

gcc -o myfile2.exe simpletron.o loader.o execute.o util.o

simpletron.o:simpletron.c simple.h

gcc -c simpletron.c

loader.o:loader.c simple.h

gcc -c loader.c

execute.o:execute.c simple.h

gcc -c execute.c

util.o:util.c simple.h

gcc -c util.c

libutil.so:util.o

gcc -shared -fPIC -o libutil.so util.o

**input.txt:**

1009

1010

2009

3110

4107

1109

4300

1110

4300

0000

0000

99999

**Error:**