#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

int load(int \*loadmemory);

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

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

int main(void)

{

int ac=0;

int ir=0;

int memory[SIZE]={0};

int ic=0;

int oc=0;

int ope=0;

int i;

    load(memory);

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

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

return 0;

}

int load(int \*loadmemory)

{

int ins;

int i=0;

printf("\*\*\* Welcome to Simpletron \*\*\*\n");

printf("\*\*\* Please enter your program one instruction \*\*\*\n");

printf("\*\*\* (or data word) at a time.I will type the  \*\*\*\n");

printf("\*\*\* location number and a question mark(?)    \*\*\*\n");

printf("\*\*\* You then type the word for that location. \*\*\*\n");

printf("\*\*\* Type the sentinel -99999 to stop entering \*\*\*\n");

printf("\*\*\* Your program \*\*\*\n");

printf("00 ? ");

scanf("%d",&ins);

while(ins!=-99999)

{

loadmemory[i]=ins;

i++;

printf("%0.2d ? ",i);

scanf("%d",&ins);

}

printf("\*\*\* Program loading completed \*\*\*\n");

printf("\*\*\* Program execution begins  \*\*\*\n");

return 0;

}

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;

}

int dump(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;

}