*//Loc Nguyen*

*// BenneTT Lawrenz*

#include <iostream>

**using** **namespace** std;

**int** temp,c;

**void** one(){

cout<<"The size of RAM = 32GB RAM\n";

}

**void** zero(){

cout<<"The size of RAM = 16GB RAM\n";

}

**void** two(){

cout<<"The size of RAM = 48GB RAM\n";

}

**void** three(){

cout<<"The size of RAM = 64GB RAM\n";

}

**void** (\*oneP)()= one;

**void** (\*zeroP)()= zero;

**void** (\*twoP)()= two;

**void** (\*threeP)()= three;

**void** checkRam(){

**\_\_asm**{

shr ax, 3;

mov c, 1;

**and** c,ax;

cmp c, 1;

je equal\_01;

equal\_00:

shr ax,1;

mov c,1;

**and** c,ax;

je second\_10;

second\_00:

call zeroP;

jmp \_restart;

second\_10:

call twoP

jmp \_restart;

equal\_01:

shr ax, 1;

mov c,1;

**and** c,ax;

cmp c,1;

je second\_11;

second\_01:

call oneP;

jmp \_restart;

second\_11:

call threeP

jmp \_restart;

\_restart:

mov ax,1100111010011100b;

}

}

**void** oneDevice(){

cout<<"There is one drive\n";

}

**void** twoDevice(){

cout<<"There are two drives\n";

}

**void** threeDevice(){

cout<<"There are three drives\n";

}

**void** fourDevice(){

cout<<"There is zero drive\n";

}

**void** (\*fourDriveP)()=fourDevice;

**void** (\*oneDriveP)()=oneDevice;

**void** (\*twoDriveP)()=twoDevice;

**void** (\*threeDriveP)()=threeDevice;

**void** checkFloopy(){

**\_\_asm**{

shr ax, 7;

mov c, 1;

**and** c,ax;

cmp c, 1;

je equal\_01;

equal\_00:

shr ax,1;

mov c,1;

**and** c,ax;

je second\_10;

second\_00:

call oneDriveP;

jmp \_restart;

second\_10:

call threeDriveP

jmp \_restart;

equal\_01:

shr ax, 1;

mov c,1;

**and** c,ax;

cmp c,1;

je second\_11;

second\_01:

call twoDriveP;

jmp \_restart;

second\_11:

call fourDriveP

jmp \_restart;

\_restart:

mov ax,1100111010011100b;

}

}

**void** zeroPrinter(){

cout<<"There is zero printer connected to computer\n";

}

**void** onePrinter(){

cout<<"There is one printer connected to computer\n";

}

**void** twoPrinter(){

cout<<"There are two printers connected to computer\n";

}

**void** threePrinter(){

cout<<"There are three printers connected to computer\n";

}

**void** (\*zeroPrinterP)()=zeroPrinter;

**void** (\*onePrinterP)()=onePrinter;

**void** (\*twoPrinterP)()=twoPrinter;

**void** (\*threePrinterP)()=threePrinter;

**void** checkPrinter(){

**\_\_asm**{

shr ax, 14;

mov c, 1;

**and** c,ax;

cmp c, 1;

je equal\_01;

equal\_00:

shr ax,1;

mov c,1;

**and** c,ax;

je second\_10;

second\_00:

call zeroPrinterP;

jmp \_restart;

second\_10:

call twoPrinterP

jmp \_restart;

equal\_01:

shr ax, 1;

mov c,1;

**and** c,ax;

cmp c,1;

je second\_11;

second\_01:

call onePrinterP;

jmp \_restart;

second\_11:

call threePrinterP

jmp \_restart;

\_restart:

mov ax,1100111010011100b;

}

}

**void** (\*checkRamP)()= checkRam;

**void** (\*checkFloopyP)()=checkFloopy;

**void** (\*checkPrinterP)()=checkPrinter;

**int** main()

{

**\_\_asm**{

mov ax, 1100111010011100b;

call checkPrinterP;

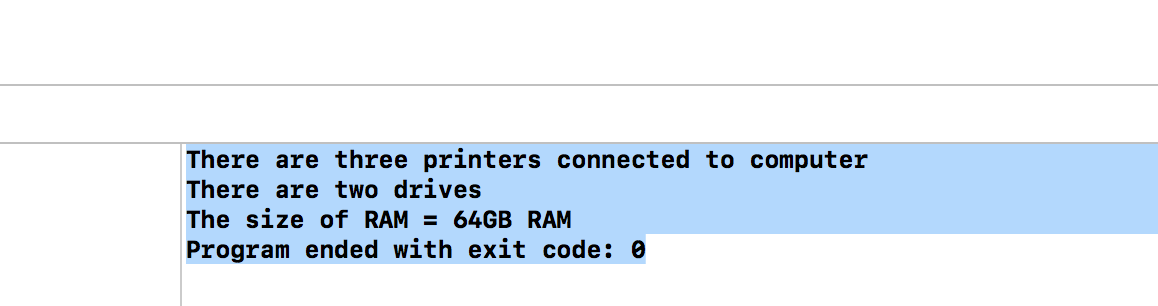
call checkFloopyP;

call checkRamP;

}

**return** 0;

}



#2

*//Loc Nguyen*

*//Bennett Lawrenz*

#include <iostream>

**using** **namespace** std;

**short** pin1= 0xBEEF,pin2=0xFADE,pin3= 0xCABE,pinTemp;

**short** total,c,two=2;

**void** validID(){

cout<<" is valid ID for the family\n";

}

**void** notValidID(){

cout<<" is not a valid ID for the family\n";

}

**void** (\*validIDP)()=validID;

**void** (\*notValidIDP)()=notValidID;

**void** checkPin(**short** pin){

pinTemp=pin;

**\_\_asm**{

mov ax, pinTemp;

mov c,0x000f;

**and** c,ax;

mov bx,0;

add bx,c;

shr ax, 4;

mov c,0x000f;

**and** c,ax;

add bx,c;

shr ax, 4;

mov c,0x000f;

**and** c,ax;

add bx,c;

shr ax, 4;

mov c,0x000f;

**and** c,ax;

add bx,c;

mov total, bx;

}

}

**void** checkValid(**short** t){

total = t;

**\_\_asm**{

mov ax, total;

cwd

idiv two;

cmp dx,1;

je notavalidID;

avalidID:

call validIDP;

jmp cont;

notavalidID:

call notValidIDP;

jmp cont;

cont:

}

}

**int** main(){

cout<<hex<<pin1;

checkPin(pin1);

checkValid(total);

cout<<hex<<pin2;

checkPin(pin2);

checkValid(total);

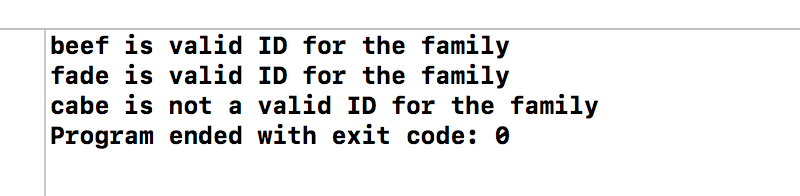
cout<<hex<<pin3;

checkPin(pin3);

checkValid(total);

**return** 0;

}



#3

#include <iostream>

#include<string>

**using** **namespace** std;

string a,b,f;

**short** c ,x,first,second,third,fourth ,temp,stage,eight=8,four=4,two=2,zero=0;

**void** addNumber(){

a+=to\_string(c);

f+=to\_string(c);

}

**void** printZero(){

b+= to\_string (stage)+ " ";

}

**void** addSpace(){

a+=" ";

}

**void** converToShort(){

zero=stoi(f);

}

**void** (\*printZeroP)()=printZero;

**void** (\*addSpaceP)()=addSpace;

**void** (\*addNumberP)()=addNumber;

**void** convertBinary(){

**\_\_asm**{

cwd

idiv eight;

mov temp,dx;

mov c,ax;

call addNumberP;

mov ax,temp;

cwd

idiv four;

mov temp,dx;

mov c,ax;

call addNumberP;

mov ax, temp;

cwd

idiv two;

mov temp,dx;

mov c,ax;

call addNumberP;

mov ax,temp;

mov c,ax;

call addNumberP;

cont:

}

}

**void** (\*convertBinaryP)()= convertBinary;

**void** numberOf1(){

**\_\_asm**{

mov ax,0x6A2F;

mov temp,0;

mov c,1;

mov bx,0;

loop:

cmp bx,16;

jg done;

**and** c,ax;

cmp c,1;

je add\_1;

jmp cont;

add\_1:

inc temp;

cont:

inc bx;

shr ax,1;

mov c,1;

jmp loop;

done:

}

}

**void** (\*numberOf1P)()=numberOf1;

**void** numberOf0(){

**\_\_asm**{

}

}

**void** (\*numberOf0P)()=numberOf0;

**int** main(){

**\_\_asm**{

mov ax, 0x6A2F;

mov fourth, 0x000F;

**and** fourth,ax;

shr ax,4;

mov third, 0x000F;

**and** third,ax;

shr ax,4;

mov second, 0x000F;

**and** second, ax;

shr ax, 4;

mov first, 0x000F;

**and** first,ax;

mov ax, first;

call convertBinaryP;

call addSpaceP;

mov ax, second;

call convertBinaryP;

call addSpaceP;

mov ax,third;

call convertBinaryP;

call addSpaceP;

mov ax,fourth;

call convertBinaryP;

call numberOf1P;

mov ax,0110101000101111b;

mov bx,1;

repeat:

cmp bx,16;

jg done;

mov x,1;

**and** x,ax;

cmp x,0;

je add\_0;

jmp increase;

add\_0:

mov stage,bx;

call printZeroP;

increase:

inc bx;

shr ax,1;

jmp repeat;

done:

}

cout<<"AX= "<<a<<endl;

cout<<"Number of working sprinkler:"<< temp<<endl;

cout<<b<<endl;

**return** 0;

}

