Trabajo Practico Nº2

Nombre: Santiago vietto

Materia: Arquitectura de computadoras 2

Docente: Ignacio Fazio

Clave: 1802890

Año: 2020

Informacion:

La placa de Desarrollo cuenta con tres puertos multipropósitos para comunicarse al exterior. Estos puertos son de 8 bits y están mapeados en memoria en los siguientes registros que se detallan a continuación:

* Registro A: registro de Entrada y Salida
* Registro B: registro solamente Entrada
* Registro C: registro solamente Salida

Se definen las siguientes funciones para poder accedes a los registros

Int LeerRegistro (int Puerto);

Void EscribirRegistro (int Puerto,int Dato);

Ejercicio 1:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **B7** | **B6** | **B5** | **B4** | **B3** | **B2** | **B1** | **B0** |
| **Puerto A** |  |  |  |  |  |  |  |  |
| **Puerto B** |  |  |  | F1 | M1 | S3 | S2 | S1 |
| **Puerto C** | C3 | C2 | C1 |  |  |  |  |  |

**/\* Funciones de respaldo \*/**

bool getBit(char puerto, int bitPos){ //Funcion para leer estado y tener el bit

char mask = 0x1;

puerto = puerto >> bitPos;

puerto = puerto & mask;

return puerto;

}

char bit\_HI(char puerto, int bitPos){ //Funcion High

char mask = 0x1;

mask = mask << bitPos;

return puerto|mask;

}

char bit\_LO(char puerto, int bitPos){ //Funcion Low

char mask = 0xFE;

mask = mask << bitPos;

return puerto&mask;

}

**/\* Solucion \*/**

char R1 = leerRegistro(1); //Registro R1 es puerto B

char R2 = leerRegistro(2); //Registro R2 es puerto C

int S1 = getBit(R1, 0);

int S2 = getBit(R1, 1);

int S3 = getBit(R1, 2);

int M1 = getBit(R1, 3);

int F1 = getBit(R1, 4);

if( S1 == 1 || M1 == 1){ //Si S1 se acciona o M1 se activa

R2 = bit\_HI( R2, 5); //Se enciende el circuito C1

}else{

R2 = bit\_LO( R2, 5); //Caso contrario se apaga

}

if( S2 == 1 || F1 == 1){ //Si S2 se acciona o F1 se activa

R2 = bit\_HI( R2, 6); //Se enciende el circuito C2

}else{

R2 = bit\_LO( R2, 6); //Caso contrario se apaga

}

if( S3 == 1 ){ //Si S3 se acciona

R2 = bit\_HI( R2, 7); //Se enciende el circuito C3

}else{

R2 = bit\_LO( R2, 7); //Caso contrario se apaga

}

escribirRegistro( 2, R2); //Actualizamos el registro con el puerto y los datos de R2

Ejercicio 2:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **B7** | **B6** | **B5** | **B4** | **B3** | **B2** | **B1** | **B0** |
| **Puerto A** |  |  |  |  |  |  |  |  |
| **Puerto B** |  |  |  |  |  | SH3 | SF2 | SF1 |
| **Puerto C** | M3 | M2 | M1 |  |  |  |  |  |

**/\* Funciones de respaldo \*/**

bool getBit(char puerto, int bitPos){ //Funcion para leer estado y tener el bit

char mask = 0x1;

puerto = puerto >> bitPos;

puerto = puerto & mask;

return puerto;

}

char bit\_HI(char puerto, int bitPos){ //Funcion High

char mask = 0x1;

mask = mask << bitPos;

return puerto|mask;

}

char bit\_LO(char puerto, int bitPos){ //Funcion Low

char mask = 0xFE;

mask = mask << bitPos;

return puerto&mask;

}

**/\* Solucion \*/**

char R1 = leerRegistro(1);

char R2 = leerRegistro(2);

int SF1 = getBit( R1, 0);

int SF2 = getBit( R1, 1);

int SH3 = getBit( R1, 2);

bool M2yM3\_encendidas = true; //Creo bandera para informar si M2 y M3 estan High

if( SH3 == 0 && SF1 = 1){ //Si SH3 indica SECO y SF1 indique LLENO

R2 = bit\_HI( R2, 7); //M3 se enciende

}else{

R2 = bit\_LO( R2, 7); //Si SF1 esta VACIO M3 se apaga, lo mismo con SH3 HUMEDO

M2yM3\_encendidas = false;

}

if( SF2 == 0 && SF1 = 1){ //Si SF2 indica VACIO y SF1 indique LLENO

R2 = bit\_HI( R2, 6); //M2 se enciende

}else{

R2 = bit\_LO( R2, 6); //Si SF1 esta VACIO M2 se apaga, lo mismo con SF2 LLENO

M2yM3\_encendidas = false;

}

if( SF1 == 0 || M2yM3\_encendidas){ //Si SF1 indica VACIO o M2 y M3 estan high

R2 = bit\_HI( R2, 7); //M1 se enciende

}else{

R2 = bit\_LO( R2, 5); //Si SF1 esta LLENO M3 se apaga, o si M2 y M3 estan low

}

escribirRegistro( 2, R2); //Actualizamos el registro con el puerto y los datos de R2

Ejercicio 3:

**/\* Funciones de respaldo \*/**

bool getBit(char puerto, int bitPos){ //Funcion para leer estado y tener el bit

char mask = 0x1;

puerto = puerto >> bitPos;

puerto = puerto & mask;

return puerto;

}

char bit\_HI(char puerto, int bitPos){ //Funcion High

char mask = 0x1;

mask = mask << bitPos;

return puerto|mask;

}

char bit\_LO(char puerto, int bitPos){ //Funcion Low

char mask = 0xFE;

mask = mask << bitPos;

return puerto&mask;

}

**/\* Solucion \*/**

char R0 = leerRegistros(0); //Trabajamos sobre el registro A

char R1 = leerRegistros(1); //Los bits mas significativos estan en el registro B

//Agarramos los bits mas significativos

int ledsAPrender = getBit(R1, 5) + getBit(R1, 6)\*2 + getBit(R1, 7)\*4;

for(int i = 0; i<8 ; i++){ //Empezamos de los menos significativos

if(ledsAPrender>0){

R0 = bit\_HI( R0, i);

ledsAPrender--;

}else{

R0 = bit\_LO( R0, i):

}

}

escribirRegistro( 0, R0);

Ejercicio 4:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **C** | | | | | | | |
|  | **C7** | **C6** | **C5** | **C4** | **C3** | **C2** | **C1** | **C0** |
| **Extractor de aire** |  |  |  |  |  |  |  | 0/1 |
| **Aire acondicionado** |  |  |  |  |  |  | 0/1 |  |
| **Calefactor electrico** |  |  |  |  |  | 0/1 |  |  |
| **Ventilador** |  |  |  |  | 0/1 |  |  |  |

\_ Usamos todo el puerto C

**/\* Funciones de respaldo \*/**

bool getBit(char puerto, int bitPos){ //Funcion para leer estado y tener el bit

char mask = 0x1;

puerto = puerto >> bitPos;

puerto = puerto & mask;

return puerto;

}

char bit\_HI(char puerto, int bitPos){ //Funcion High

char mask = 0x1;

mask = mask << bitPos;

return puerto|mask;

}

char bit\_LO(char puerto, int bitPos){ //Funcion Low

char mask = 0xFE;

mask = mask << bitPos;

return puerto&mask;

}

**/\* Solucion \*/**

char R1 = leerRegistro(1);

char R2 = leerRegistro(2); //Trabajamos sobre el registro C

int temperatura = R1;

if(temperatura < 22){ //Con temperatura menor a 22

R2 = 0B00001100 //Se enciende el calefactor electrico y el ventilador

/\*Otra forma:

R2 = bit\_HI( R2, 2);

R2 = bit\_HI( R2, 3); \*/

}else{ //Con temperatura mayor a 22

if(temperatura>26 && temperatura<80){

R2 = 0b00000001; //Se enciende el extractor

/\*Otra forma:

R2 = bit\_HI( R2, 0); \*/

}else{

if(temperatura >= 28){

R2 = 0b00000010; //Se apaga el extractor y se enciende el aire acondionado

/\*Otra forma:

R2 = bit\_HI( R2, 1);

R2 = bit\_LO( R2, 0); \*/

}else{ //Si la temperatura esta entre 22 y 26

R2 = 0b00000000; //No funciona ningun acturador

}

}

}

escribirRegistro( 2, R2);

Ejercicio 5:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **B7** | **B6** | **B5** | **B4** | **B3** | **B2** | **B1** | **B0** |
| **Puerto A** |  |  |  |  |  |  |  |  |
| **Puerto B** |  |  |  | C1 | Smdo | Smup | S2 | S1 |
| **Puerto C** |  |  |  |  | Lroja | Lverde | M2 | M1 |

**/\* Funciones de respaldo \*/**

bool getBit(char puerto, int bitPos){ //Funcion para leer estado y tener el bit

char mask = 0x1;

puerto = puerto >> bitPos;

puerto = puerto & mask;

return puerto;

}

char bit\_HI(char puerto, int bitPos){ //Funcion High

char mask = 0x1;

mask = mask << bitPos;

return puerto|mask;

}

char bit\_LO(char puerto, int bitPos){ //Funcion Low

char mask = 0xFE;

mask = mask << bitPos;

return puerto&mask;

}

**/\* Solucion \*/**

char R1 = leerRegistro(1);

char R2 = leerRegistro(2);

//Inputs

int C1 = getBit( R1, 4);

int Smdo = getBit( R1, 3);

int Smup = getBit( R1, 2);

int S2 = getBit( R1, 1);

int S1 = getBit( R1, 0);

//Outputs

int M1 = getBit( R1, 0);

int M2 = getBit( R1, 1);

if(S1 == 1 && C1 == 1 && Smup == 0 && M2 == 0){

R2 = bit\_HI( R2, 0); //Se enciende M1

}

if( Smup == 1 ){

R2 = bit\_LO( R2, 0); //Se apaga M1;

}

if( S2 == 1 && Smd0 == 0 && M1 == 0){

sleep(30); //Se enciende M2 por 30 segundos

R2 = bit\_LO( R2, 1);

}

if( Smdo == 1 ){

R2 = bit\_LO( R2, 1); //Se apaga M2;

}

if( S1 == 0 && S2 == 0){

R2 = bit\_LO( R2, 3); //Se apaga luz roja

R2 = bit\_HI( R2, 2); //Se enciende luz verde

}else{

R2 = bit\_HI( R2, 3); //Se enciende luz roja

R2 = bit\_LO( R2, 2); //Se apaga luz verde

}

escribirRegistro( 2, R2);