

# Objetivo

Demostrar el conocimiento necesario para implementar y ejecutar un código en lenguaje c y ensamblador, así mismo acceder a los puertos de la tarjeta T-Juino por medio de las bibliotecas.

# Desarrollo

Realice lo siguiente.

1. Tomando de ejemplo el archivo main.c

|  |
| --- |
| #include <stdio.h> extern unsigned int \_test (unsigned int, unsigned int);  int main(void)  { printf("%d\n", \_test (85,5)); /\* you are calling div here, not \_test \*/ return 0;  } |

1. Archivo fun.asm

global \_test

\_test:

push ebp mov ebp, esp mov eax, [ebp+8] xor edx, edx div dword [ebp+12] mov esp, ebp pop ebp ret

1. Correr en terminar las siguientes líneas

nasm -f elf -o fun.o fun.asm gcc -m32 -c -o main.o main.c gcc -m32 -o run fun.o main.o

./run

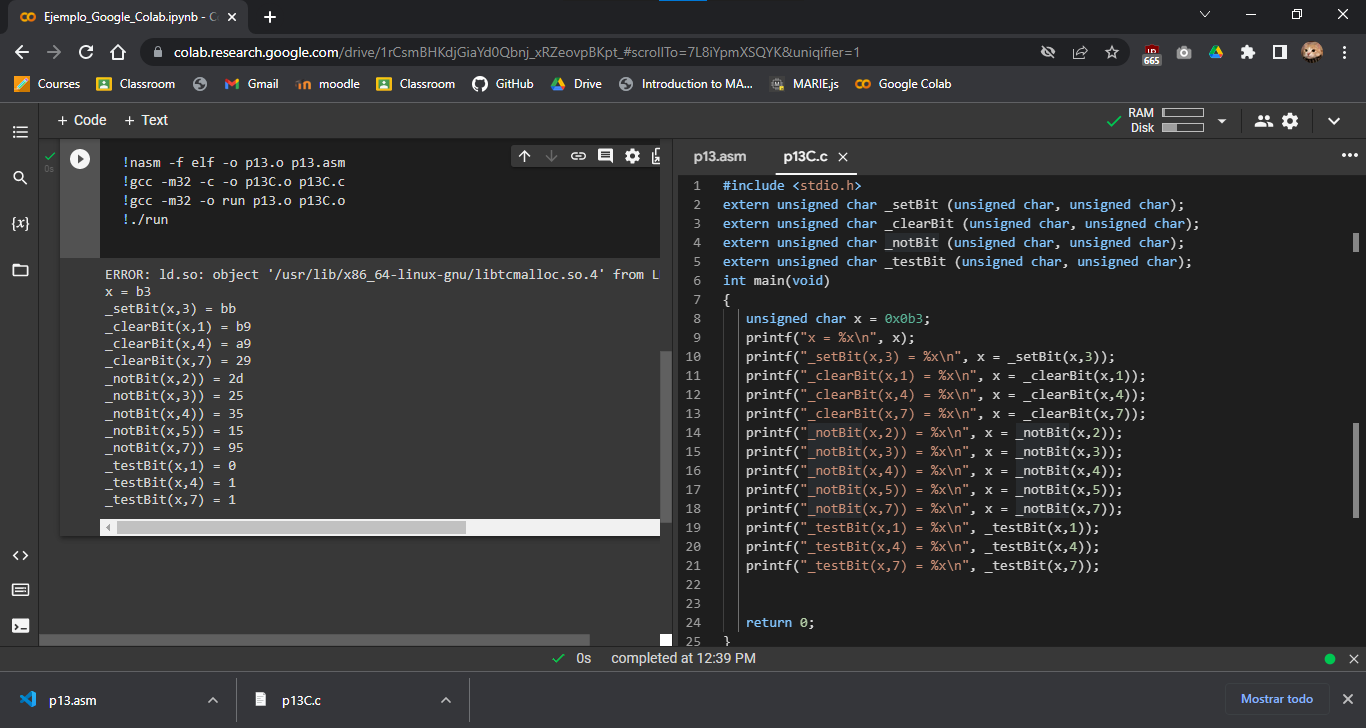
1. Agregar captura de pantalla del resultado de la ejecución impreso en terminal.
2. Apoyándose en el ejemplo anterior crear un archivo P13.asm donde estarán las subrutinas y el código necesario para hacer el llamado a estas desde lenguaje c:
3. **setBit**: activa un bit del 0 al 7 a un dato de 1 Byte recibidos ambos por parámetro, está retorna únicamente el dato modificado.
4. **clearBit**: desactiva un bit del 0 al 7 a un dato de 1 Byte recibidos ambos por parámetro, está retorna únicamente el dato modificado.
5. **notBit**: invierte un bit del 0 al 7 a un dato de 1 Byte recibidos ambos por parámetro, está retorna únicamente el dato modificado.
6. **testBit**: revisa el estado de un bit del 0 al 7 a un dato de 1 Byte recibidos por parámetro, está retorna únicamente el valor del bit.

6. Crear un archivo P13C.c donde se llamarán a las funciones de ensamblador como funciones externas.

1. Se les enviará el dato que será modificado de un Byte y el número del bit en el que se trabajara.
2. Para probar el funcionamiento se deberá hacer lo siguiente: c)

|  |  |
| --- | --- |
| 1. | Comenzar con un dato que será 0B3h al cual se le debe activar el bit 3. |
| 2. | El dato resultante del paso 1 desactivarle los bits 1, 4 y 7. |
| 3. | El dato resultante del paso 2 invertirle los bits 2, 3, 4, 5 y 7. |
| 4. | El dato resultante del paso 3 revisar el estado de los bits 1, 4 y 7. |

1. Mostrar en terminar como hexadecimal cada cambio que se va realizando por cada modificación de bit.



# Conclusiones y comentarios Dificultades en el desarrollo Referencias

Página 2