REGISTROS DE CONTROL Y ESTADO

Para discutir y comprender el funcionamiento del procesador, podemos considerar los requisitos que tiene que cumplir:

1. Buscar instrucciones desde la memoria o desde un dispositivo de entrada y salida

2. Decodificar instrucciones para determinar que acción es necesaria

3. Buscar los datos que la ejecución de la instrucción puede requerir

4. Procesar los datos que la ejecución puede necesitar a través de una operación lógica o aritmética

5. Almacenar los resultados donde corresponda

Para hacer esas cosas el procesador debe:

Almacenar algunos datos temporalmente.

Recordar la posición de la ultima instrucción de forma de poder determinar de donde tomar la siguiente.

Necesita almacenar instrucciones y datos temporalmente mientras una instrucción está ejecutándose.

En otras palabras, el procesador necesita una pequeña memoria interna.

Dentro del procesador hay un conjunto de registros clasificados en dos tipos:

Registros visibles al programador:	**Registros de control y estado:**
Permiten al programador de lenguaje máquina o de ensamblador minimizar las referencias a memoria principal por medio de la optimización de uso de registros	Son utilizados por la unidad de control para controlar el funcionamiento del procesador y por programas privilegiados del sistema operativo para controlar la ejecución de programas.

REGISTROS DE CONTROL Y ESTADO

Contador de programa (Program Counter- PC):	Registro de instrucción (Instruction Register- IR):	Registro de dirección de memoria (Memory Address Register- MAR)	Memory Buffer Register - MBR
Contiene la dirección de la instrucción a buscar	Contiene la instrucción buscada mas recientemente	Contiene la dirección de una posición de memoria Registro amortiguador de memoria	Contiene el dato a escribir en una posición de memoria o el dato contenido en una posición de memoria leído mas recientemente

No todos los procesadores tienen registros internos designados como MAR o MBR pero se necesita algún mecanismo de almacenamiento intermedio equivalente mediante el cual se de salida a los bits que van a ser transferidos por el bus de sistema se almacenen los bits leídos por el bus de datos.

PC

Típicamente, el procesador actualiza el PC luego de cada búsqueda de instrucción, de manera que siempre apunte a la siguiente instrucción a ejecutar.

Una instrucción de bifurcación o salto también modificara el valor del registro PC.

IR

La instrucción buscada se carga en IR, donde son analizados los códigos de operación y los campos de operando.

O SEA CONTIENE LA INSTRUCCIÓN QUE SE ESTÁ EJECUTANDO

MBR Y MAR

Se intercambian datos con la memoria
por medio del MAR y el MBR.
El MAR se conecta
con el bus de direcciones y
el MBR con el
bus de datos.
Los registros visibles por el usuario
repetidamente cambian datos con MBR.

MODULARIZACIÓN

REUSO

LLAMADO A RUTINAS:

CALL Y RET

RUTINA

ES UN PROGRAMA QUE RESUELVE UN PROBLEMA ACOTADO Y RECURRENTE,
PERMITE REUTILIZAR LA IDEA DE LA SOLUCIÓN Y ADEMÁS PERMITE PARTIR EL PROBLEMA (MODULARIZACIÓN)

TAMBIÉN SE PUEDE LLAMAR SUBRUTINA, PORQUE SE UTILIZA COMO PARTE DE UN PGM MÁS GRANDE

MODULARIZAR:

DIVIDIR EL PROBLEMA GRANDE EN PEQUEÑOS

REUSAR:

PENSAR UNA SOLUCIÓN PARA QUE SE ADAPTE A DISTINTAS SITUACIONES

- **¿CÓMO SE INTEGRAN LAS PARTES O RUTINAS?**

Se necesitan hacer 2 cosas

- 1. ENCAPSULAR RUTINAS:

NECESITAMOS UNA ETIQUETA QUE LA IDENTIFIQUE Y UN FIN O INSTRUCCIÓN RET

Ej:

```
sumatres:  MOV R1, R0
           ADD R1, 0X0003
           RET
```

- **2.LLAMAR LAS SUBRUTINAS PARA USARLAS:**

HACIENDO REFERENCIA A LA ETIQUETA QUE LA IDENTIFICA:

Ej.:

CALL sumatres

EFECTO DE CALL Y RET

- CALL: hace que la UC altere el flujo del programa, es decir que se desvíe a la rutina que llama, y de esa forma cuando finaliza pueda volver(guarda la dirección a donde tiene que volver)

- RET: Restituye el flujo del programa a la instrucción siguiente al último llamado (a la dirección que guardó el CALL

- DOCUMENTAR:

¿Cómo documentar el código?

• **Requiere**	Qué necesita la rutina (Parámetros y precondiciones) ¿Dónde están los parámetros? (en que variables) ¿Que características deben tener? (distinto de 0, etc)
• **Retorna**	En que variable (registro o memoria) se retorna el resultado
• **Modifica**	Que variables auxiliares se utilizan (registros, memoria, flags)

REQUIERE: VALOR AL QUE SE DESEA SUMARLE TRES ALMACENADO EN RO

MODIFICA: R1

RETORNA: EN R1 EL RESULTADO DE SUMARLE TRES AL NÚMERO ALMACENADO EN R0

- PILA

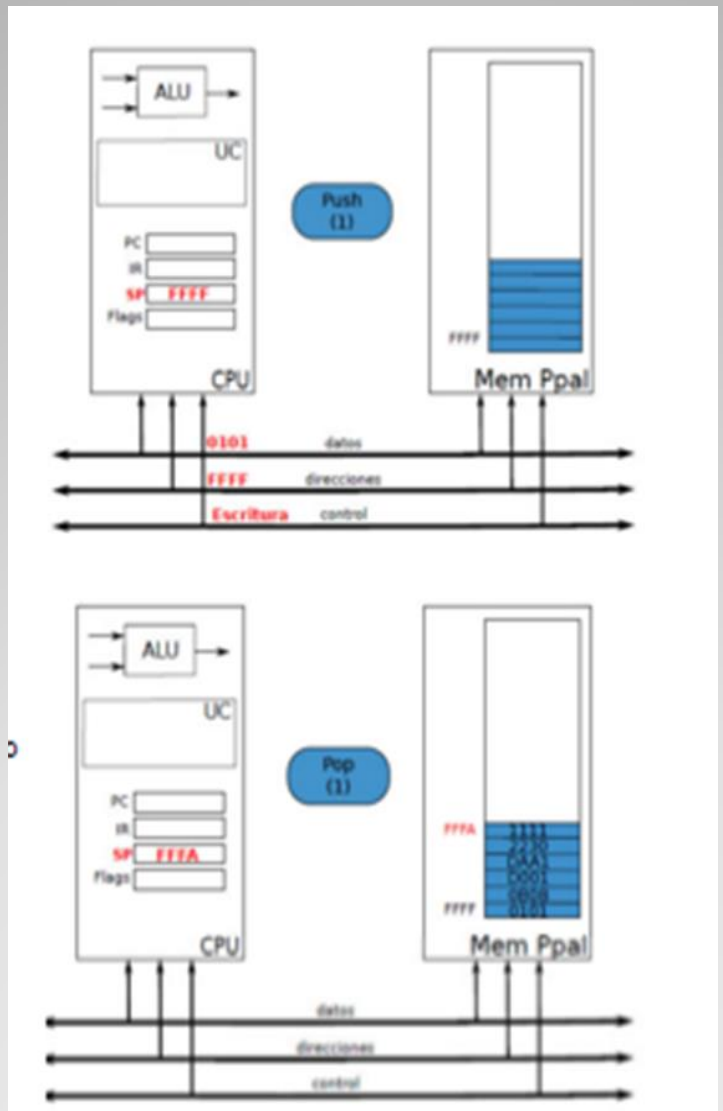
ESTRUCTURA PARA ALMACENAR DATOS

- PUSH

- PONEMOS UN PLATO EN LA PILA(APILAMOS)
- AGREGAMOS UN ELEMENTO SOBRE EL ÚTLIMO AGREGADO

- POP

- RETIRAMOS UN PLATO DE LA PILA(DESAPILAMOS)
- SACAMOS DEL TOPE O PRIMER PLATO DE LA PILA


ALU
UC
PC
IR
SP
FFFF
Flags
CPU
Push
(1)
FFFF
Mem Ppal
0101
datos
FFFF
direcciones
Escritura
control
ALU
UC
PC
IR
SP
FFFA
Flags
CPU
Pop
(1)
FFFA
FFFF
Mem Ppal
datos
direcciones
control

Asumamos que la pila arranca en la dirección FFEF DE MEMORIA

Qué pasa si queremos apilar el dato 0101 en la pila??????

El tope de la pila va a pasar a ser FFEE, o sea que

CON CADA PUSH SE DECREMENTA EL VALOR EN 1

Si queremos hacer un POP (desapilar),
el tope de la pila debe reajustarse y
luego copiarse este dato,
porque el tope de la pila representa
siempre el primer lugar disponible sobre los
valores ya apilados

ENTONCES EL VALOR DEL TOPE DE PILA SE INCREMENTA

SIEMPRE DEBEMOS SABER DONDE ESTA EL TOPE DE LA PILA

PARA EL SEGUIMIENTO DE ESTE TOPE UTILIZAMOS EL:

SP
STACK POINTER

SP

ESTE REGISTRO CONTIENE LA DIRECCIÓN DE LA PRIMERA CELDA DE MEMORIA DISPONIBLE DE LA PILA

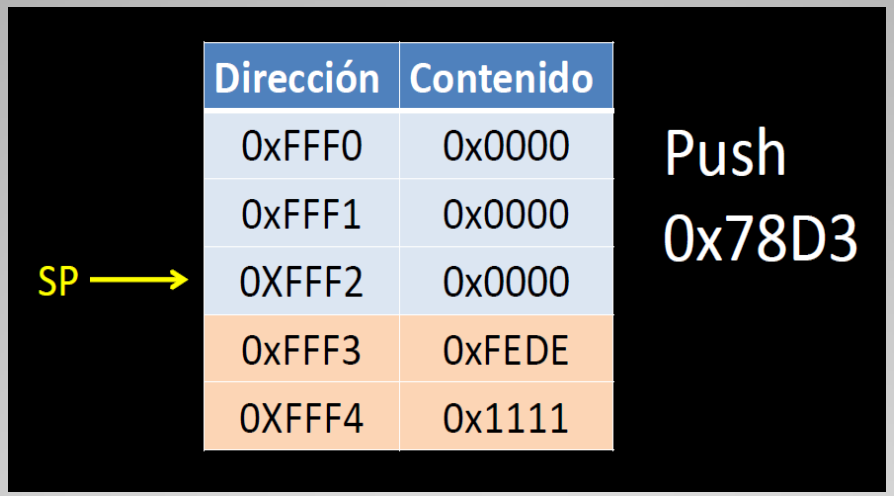
- ENTONCES….

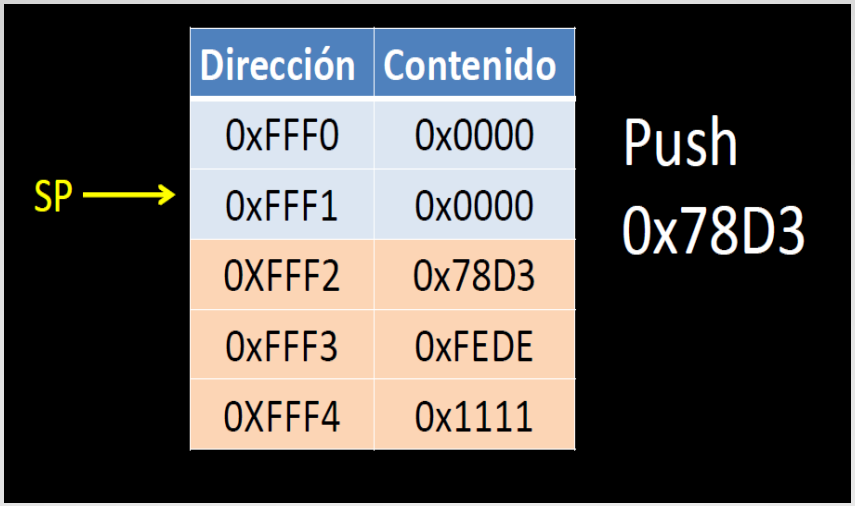
SI HAGO PUSH:

SE HACE ESCRITURA DEL DATO QUE ESTÁ EN EL BUS DE DATOS EN LA DIRECCIÓN QUE ESTÁ EL SP, PARA DEJAR TODO LISTO AL NUEVO TOPE DE PILA.


Dirección
Contenido
0xFFF0
0x0000
0xFFF1
0x0000
0XFFF2
0x0000
SP
0xFFF3
0x0A53
0XFFF4
0x1111
Push
0xFEDE


Dirección
Contenido
0xFFF0
0x0000
0xFFF1
0x0000
SP
0XFFF2
0x0000
0xFFF3
0xFEDE
0XFFF4
0x1111
Push
0xFEDE


Dirección
Contenido
0xFFF0
0x0000
0xFFF1
0x0000
SP
0XFFF2
0x0000
0xFFF3
0xFEDE
0XFFF4
0x1111
Push
0x78D3


Dirección
Contenido
0xFFF0
0x0000
SP
0xFFF1
0x0000
0XFFF2
0x78D3
0xFFF3
0xFEDE
0XFFF4
0x1111
Push
0x78D3

- SI HAGO POP, SE INCREMENTA EL SP, SE HACE UNA LECTURA DE LA DIRECCIÓN QUE ESTÁ EN EL SP

- TENGAMOS EN CUENTA QUE:

EL TAMAÑO Y UBICACIÓN DE LA PILA ESTÁ DEFINIDO POR LA ARQUITECTURA

EN Q3 COMIENZA EN FFFE

POP NO BLANQUEA EL TOPE DE PILA, SINO QUE NO SE PODRÁ ACCEDER

- Relacionamos lo visto con CALL Y RET

- CALL: Desvía el flujo del programa a la instrucción que define la etiqueta

- RET: Permite restituir el flujo del programa a la instrucción siguiente al último llamado.

SE UTILIZA UNA PILA

CALL APILA

RET DESAPILA

Lo que apila es una dirección,

¿QUIÉN NOS DICE CUÁL ES LA SIGUIENTE INSTRUCCIÓN A EJECUTAR?

PC

PROGRAM COUNTER

- ARQUITECTURA Q3:

INSTRUCCIONES DE UN OPERANDO ORIGEN, hablar de modo de direccionamiento inmediato (las etiquetas son direcciones en definitiva)

Cod Op (4bits)	Relleno (000000)	Modo origen (6 bits)	Origen (16 bits)

Operación	Código	Efecto
CALL	1011	[SP] ←PC; SP←SP-1; PC ←Origen

INSTRUCCIONES SIN OPERANDOS

Cod Op (4bits)	Relleno (000000000000)

Operación	Código	Efecto
RET	1100	PC ←[SP+1]; SP←SP + 1