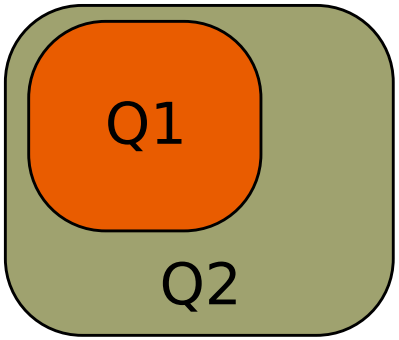
Modularización y reuso de código

Organización de computadoras

Universidad Nacional de Quilmes

11 de septiembre de 2013

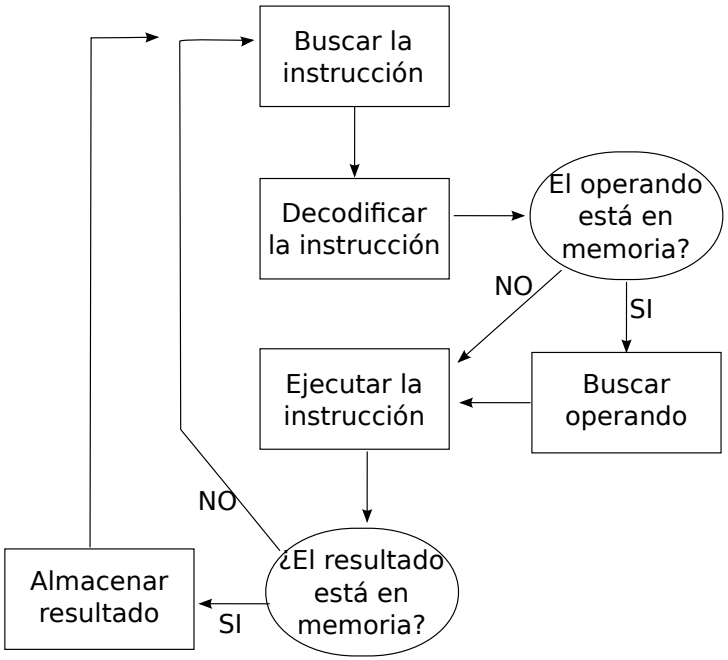
Repaso



1. Memoria
   1. celdas
   2. palabras
   3. relación con circuitos conocidos (Flip-flop, decodificador)
2. Buses
   1. de Direcciones
   2. de Datos
   3. de Control
3. **Q2**
   1. Modo de direccionamiento directo
4. Ciclo de ejecución revisado:
   1. Cantidad de accesos

Ciclo de ejecución revisado

Ciclo de ejecución revisado



¿En que parte de la memoria se busca la instrucción?

Ciclo de ejecución revisado

¿Cómo sabe la UC cuál es la celda que debe leer cada vez?

Ciclo de ejecución revisado

¿Cómo sabe la UC cuál es la celda que debe leer cada vez?

Utiliza un registro especial: Program Counter (PC)

Ciclo de ejecución revisado

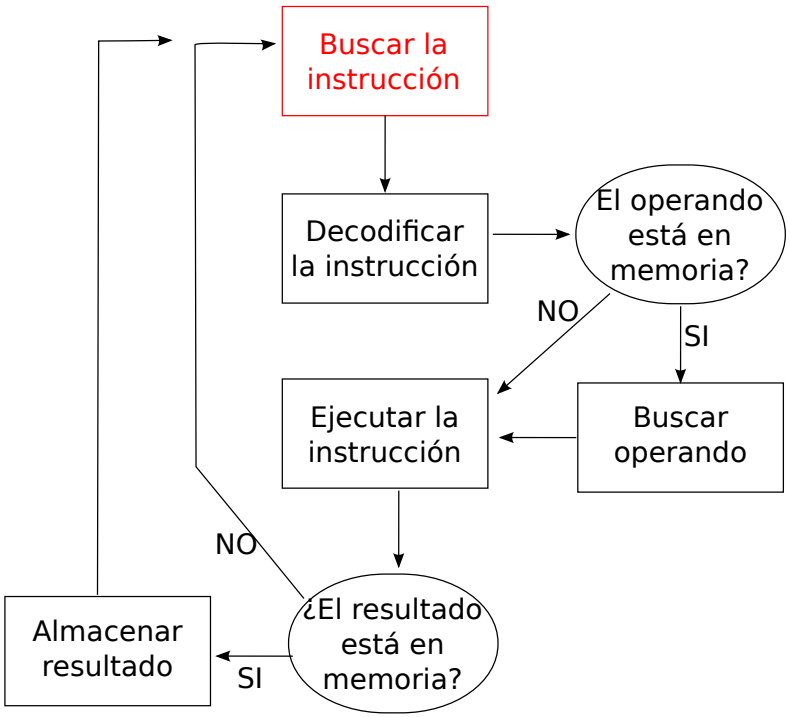
¿Dónde se almacena la instrucción leída?

Ciclo de ejecución revisado

¿Dónde se almacena la instrucción leída?

En otro registro especial: Instruction Register (IR)

Ciclo de ejecución revisado



1. Se hace una lectura de la celda de memoria que indica PC.
2. El contenido de la celda leida se carga en IR
3. Se incrementa PC en 1

Ciclo de ejecución revisado

0	0101
1	1010
2	0000
3	1111
4	1100

- La instrucción actual es 0101
- El valor de PC es 1

Registros reservados

Entonces...

PC (Program Counter) indica la dirección de la **siguiente instrucción** a ejecutar

IR (Instruction Register) Almacena el código máquina de la **instrucción actual**

Program Counter: Ejemplo

0000	1200
0001	000F
0002	1111
0003	29C8
0004	A0A0
0005	0000
0006	0000
0007	0000
0008	0000
0009	0000
000A	0000
000B	0000
000C	0000
000D	0000
000E	0000
000F	0000

1. PC=0

Program Counter: Ejemplo

0000	1200
0001	000F
0002	1111
0003	29C8
0004	A0A0
0005	0000
0006	0000
0007	0000
0008	0000
0009	0000
000A	0000
000B	0000
000C	0000
000D	0000
000E	0000
000F	0000

1. PC=0
2. Búsqueda de instrucción:
   1. Lectura de la celda 0000
   2. IR=1200
   3. PC=0001

Program Counter: Ejemplo

0000	1200
0001	000F
0002	1111
0003	29C8
0004	A0A0
0005	0000
0006	0000
0007	0000
0008	0000
0009	0000
000A	0000
000B	0000
000C	0000
000D	0000
000E	0000
000F	0000

1. PC=0
2. Búsqueda de instrucción (celda 0000)
3. Decodificación de la instrucción: MOV [??], 0x??

Program Counter: Ejemplo

0000	1200
0001	000F
0002	1111
0003	29C8
0004	A0A0
0005	0000
0006	0000
0007	0000
0008	0000
0009	0000
000A	0000
000B	0000
000C	0000
000D	0000
000E	0000
000F	0000

1. PC=0
2. Búsqueda de instrucción (celda 0000)
3. Decodificación de la instrucción: MOV [??], 0x??
4. Búsqueda de instrucción:
   1. Lectura de la celda 0001
   2. IR=1200000F
   3. PC=0002

Program Counter: Ejemplo

0000	1200
0001	000F
0002	1111
0003	29C8
0004	A0A0
0005	0000
0006	0000
0007	0000
0008	0000
0009	0000
000A	0000
000B	0000
000C	0000
000D	0000
000E	0000
000F	0000

1. PC=0
2. Búsqueda de instrucción (celda 0000)
3. Decodificación de la instrucción: MOV [??], 0x??
4. Búsqueda de instrucción (celda 0001)
5. Búsqueda de instrucción:
   1. Lectura de la celda 0002
   2. IR=1200000F0000
   3. PC=0003

Program Counter: Ejemplo

0000	1200
0001	000F
0002	1111
0003	29C8
0004	A0A0
0005	0000
0006	0000
0007	0000
0008	0000
0009	0000
000A	0000
000B	0000
000C	0000
000D	0000
000E	0000
000F	0000

1. PC=0
2. Búsqueda de instrucción (celda 0000)
3. Decodificación de la instrucción: MOV [??], 0x??
4. Búsqueda de instrucción (celda 0001)
5. Búsqueda de instrucción (celda 0002)
6. Decodificación de la instrucción: MOV [000F], 0x0000

Program Counter: Ejemplo

0000	1200
0001	000F
0002	1111
0003	29C8
0004	A0A0
0005	0000
0006	0000
0007	0000
0008	0000
0009	0000
000A	0000
000B	0000
000C	0000
000D	0000
000E	0000
000F	1111

1. PC=0
2. Búsqueda de instrucción (celda 0000)
3. Decodificación de la instrucción: MOV [??], 0x??
4. Búsqueda de instrucción (celda 0001)
5. Búsqueda de instrucción (celda 0002)
6. Decodificación de la instrucción: MOV [000F], 0x0000
7. Ejecución de la instrucción

La decodificación de la instrucción **NO** es el desensamblado

Desafíos

Desafío de programación número 1

Hacer un programa que dados calcule n^5 para los números en las celdas A001 a A003 y lo guarde en la misma celda

A001	0003
A002	0001
A003	0008

Desafío de programación número 1

Hacer un programa que dados calcule n^5 para los números en las celdas A001 a A003 y lo guarde en la misma celda

A001	0003
A002	0001
A003	0008

A001	00F3
A002	0001
A003	8000

Desafío de programación número 1

Bosquejando el programa: ¿Que hay que hacer?

Desafío de programación número 1

Bosquejando el programa: ¿Que hay que hacer?

- Copiar el contenido de 0001 a un registro
- Multiplicarlo 4 veces por si mismo
- Copiar el resultado a 0001
- Copiar el contenido de 0002 a un registro
- Multiplicarlo 4 veces por si mismo
- Copiar el resultado a 0002

Desafío de programación número 1

Bosquejando el programa: ¿Que hay que hacer?

- Copiar el contenido de 0001 a un registro
- Multiplicarlo 4 veces por si mismo
- Copiar el resultado a 0001
- Copiar el contenido de 0002 a un registro
- Multiplicarlo 4 veces por si mismo
- Copiar el resultado a 0002
- Copiar el contenido de 0003 a un registro
- Multiplicarlo 4 veces por si mismo
- Copiar el resultado a 0003

Desafío de programación número 2

Hacer un programa que dados los números (n) en las celdas 0001 a 0005 calcule

$$f(n) = (15 * n + 4)/12 - n/2$$

y lo guarde en las celdas 0006 a 000A

0001	n_1
0002	n_2
0003	n_3
0004	n_4
0005	n_5
0006	$f(n_1)$
0007	$f(n_2)$
0008	$f(n_3)$
0009	$f(n_4)$
000A	$f(n_5)$

Rutinas

Rutina

- Programa que resuelve un problema acotado

Rutinas

Rutina

- Programa que resuelve un problema acotado
- Puede ser usado en muchas ocasiones

Rutinas

Rutina

- Programa que resuelve un problema acotado
- Puede ser usado en muchas ocasiones
- Permite **modularizar** y **reusar** código

Rutinas

Rutina

- Programa que resuelve un problema acotado
- Puede ser usado en muchas ocasiones
- Permite **modularizar** y **reusar** código
- También se la llama subrutina (Es sub porque se la piensa para ser utilizada dentro de otro programa)

Modularización

Modularización

Modularizar
Dividir un problema *grande* en problemas mas *pequeños*

Modularización

Modularizar
Dividir un problema *grande* en problemas mas *pequeños*

Problemón

Problemita 1

Problemita 2

Modularización

Desafío de programación número 1

Modularización

Desafío de programación número 1

Programa principal

- Calcular potencia de A001
- Calcular potencia de A002
- Calcular potencia de A003

Modularización

Calcular potencia de A001

```
MOV R0, [A001]    ; copiar valor original
MUL R0, [A001]
MUL R0, [A001]
MUL R0, [A001]
MOV [A001],R0     ; mover resultado
```

Modularización

Calcular potencia de A002

```
MOV R0, [A002]    ; copiar valor original
MUL R0, [A002]
MUL R0, [A002]
MUL R0, [A002]
MOV [A002],R0     ; mover resultado
```

Modularización

Calcular potencia de A003

```
MOV R0, [A003]    ; copiar valor original
MUL R0, [A003]
MUL R0, [A003]
MUL R0, [A003]
MOV [A003],R0     ; mover resultado
```

¿Cómo se integran las partes?

Integrar las partes

Encapsular las subrutinas

Integrar las partes

Encapsular las subrutinas

delimitar el comienzo y el fin de la rutina

Integrar las partes

Encapsular las subrutinas

```
potA001:    MOV R0, [A001]    ; etiqueta de comienzo
            MUL R0, [A001]
            MUL R0, [A001]
            MUL R0, [A001]
            MOV [A001],R0     ;
            RET               ; fin de la rutina
```

Integrar las partes

Encapsular las subrutinas

```
potA002:    MOV R0, [A002]    ; etiqueta de comienzo
            MUL R0, [A002]
            MUL R0, [A002]
            MUL R0, [A002]
            MOV [A002],R0     ;
            RET               ; fin de la rutina
```

Integrar las partes

Encapsular las subrutinas

```
potA003:    MOV R0, [A003]    ; etiqueta de comienzo
            MUL R0, [A003]
            MUL R0, [A003]
            MUL R0, [A003]
            MOV [A003],R0     ;
            RET               ; fin de la rutina
```

Integrar las partes

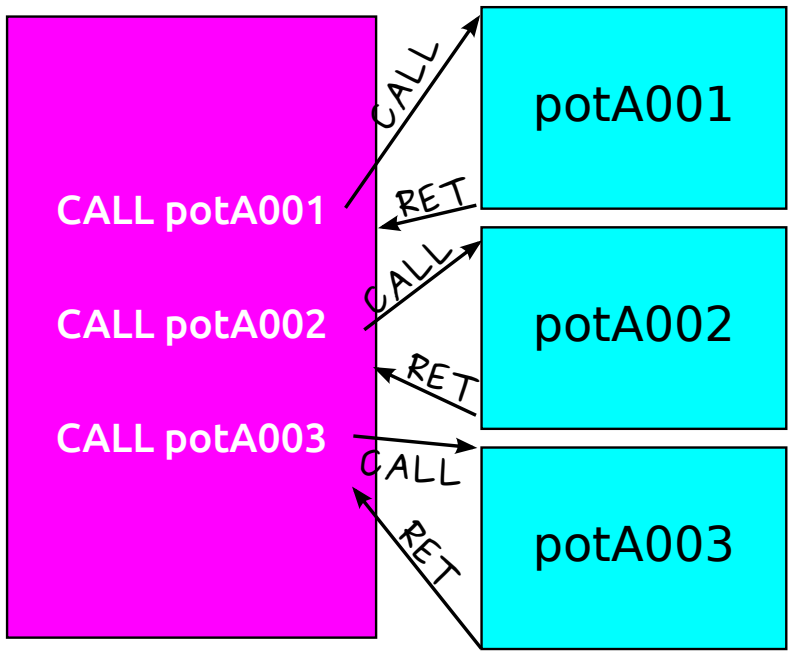
Usar (llamar) las subrutinas

```
CALL potA001
CALL potA002
CALL potA003
```

Modularización: CALL y RET

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

Instrucción RET
Permite restituir el flujo del programa a la instrucción siguiente del último llamado



¿Como puedo ahorrar trabajo?

Reuso: parámetros

¿Que hay en común?

```
MOV R0,[A001]
MUL R0,[A001]
MUL R0,[A001]
MUL R0,[A001]
MOV[A001],R0
RET
```

```
MOV R0,[A002]
MUL R0,[A002]
MUL R0,[A002]
MUL R0,[A002]
MOV[A002],R0
RET
```

```
MOV R0,[A003]
MUL R0,[A003]
MUL R0,[A003]
MUL R0,[A003]
MOV[A003],R0
RET
```

Reuso: parámetros

¿En que se diferencian?

```
MOV R0,[A001]
MUL R0,[A001]
MUL R0,[A001]
MUL R0,[A001]
MOV[A001],R0
RET
```

```
MOV R0,[A002]
MUL R0,[A002]
MUL R0,[A002]
MUL R0,[A002]
MOV[A002],R0
RET
```

```
MOV R0,[A003]
MUL R0,[A003]
MUL R0,[A003]
MUL R0,[A003]
MOV[A003],R0
RET
```

Reuso

Reuso

Reusar
Escribir subrutinas que puedan ser usadas en diferentes situaciones

Reuso

Reusar
Escribir subrutinas que puedan ser usadas en diferentes situaciones

Problema 1
- Subrutina 1
- Subrutina 2

Problema 2
- Subrutina 1
- Subrutina 3

Parámetros

Reuso: parámetros

Parámetros

```
potencia    MOV R0,???
            MUL R0,???
            MUL R0,???
            MUL R0,???
            MOV ???,R0
            RET
```

La rutina **potencia** es mas flexible

Reuso: Ejercicio

Se cuenta con una rutina que aplica un descuento dado como parámetro, sobre una valor que también se pasa como parámetro:

```
descontar:  MOV R6, R3      ; valor original
            MUL R6, R4      ; descuento a aplicar
            DIV R6, 0x0064  ; multiplicar por 100
            SUB R3, R6
            RET
```

Reuso: Ejercicio

Se cuenta con una rutina que aplica un descuento dado como parámetro, sobre una valor que también se pasa como parámetro:

```
descontar:  MOV R6, R3      ; valor original
            MUL R6, R4      ; descuento a aplicar
            DIV R6, 0x0064  ; multiplicar por 100
            SUB R3, R6
            RET
```

Hacer un programa que aplique un 10 % de descuento al valor en la celda 6565, un 30 % al valor en la celda AAAA y un 45 % al valor en 0367.

Reuso: Ejercicio

Hacer un programa que aplique un 10 % de descuento al valor en la celda 6565, un 30 % al valor en la celda AAAA y un 45 % al valor en 0367.

```
MOV R3, [6565]
MOV R4, 0x000A      ; 10 %
CALL descontar;
MOV R3, [AAAA]
MOV R4, 0x001E      ; 30 %
CALL descontar;
MOV R3, [0367]
MOV R4, 0x002D      ;45 %
CALL descontar;
```

```
descontar:  MOV R6, R3
            MUL R6, R4
            DIV R6, 0x0064
            SUB R3, R6
            RET
```

Contratos

Ejercicio

En dos grupos

Equipo A Escribir una rutina que calcule el promedio de las notas de un alumno, asumiendo que las notas están en R4 y R5. El resultado debe dejarse en R6.

Equipo B Escribir un programa que calcule el promedio de 4 estudiantes de una comisión, cuyas notas están en las celdas 0B00 a 0B07.

Inconvenientes

¿Cuál es el problema del código no documentado?

- Para entender lo que hace una rutina se debe tratar de comprender el código
- Es dificil detectar errores secundarios: Cuando la rutina modifica algo no esperado
- Es dificil modificar el código para nuevos requerimientos

Contratos

¿Cómo documentar el código?

Especificar:

- **Requiere** Qué necesita la rutina (Parámetros y precondiciones)
- **Retorna** En que variable (registro o memoria) se retorna el resultado
- **Modifica** Que variables auxiliares se utilizan (registros, memoria, flags)

Contratos

¿Cómo documentar el código?

Especificar:

Requiere Qué necesita la rutina (Parámetros y precondiciones)
¿Dónde están los parámetros? (en que variables)
¿Que carácterísticas 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)

Contratos

¿Cómo documentar el código?

Especificar:

Requiere Qué necesita la rutina (Parámetros y precondiciones)

Retorna En que variable (registro o memoria) se retorna el resultado
¿Qué características es importante marcar del resultado?

Modifica Que variables auxiliares se utilizan (registros, memoria, flags)

Contratos

¿Cómo documentar el código?

Especificar:

- **Requiere** Qué necesita la rutina (Parámetros y precondiciones)
- **Retorna** En que variable (registro o memoria) se retorna el resultado
- **Modifica** Que variables auxiliares se utilizan (registros, memoria, flags)
  ¿Cambia alguna variable que no es el resultado?
  ¿Cambian las variables que contienen parámetros?

Ejercicio: documentar la rutina **promedio**

Requiere

Retorna

Modifica

Ejercicio: documentar la rutina **promedio**

Requiere Las notas están en los registros R4 y R5. Los valores están en $BSS(16)$

Retorna El promedio en el registro R6, donde el promedio resulta de la división entera: (R4+R5) %2

Modifica nada

Ejercicio: documentar la rutina **promedio**

Requiere Las notas están en los registros R4 y R5. Los valores están en $BSS(16)$

Retorna El promedio en el registro R6, donde el promedio resulta de la división entera: (R4+R5) %2

Modifica nada

Para documentar... ¿No es necesario haberla implementado??

Ejercicio: usar la rutina **promedio** para calcular el promedio de 4 estudiantes de una comisión, cuyas notas están en las celdas 0B00 a 0B07, y ponga los resultados en las celdas 0B08 a 0B0B

Ejercicio: usar la rutina **promedio** para calcular el promedio de 4 estudiantes de una comisión, cuyas notas están en las celdas 0B00 a 0B07, y ponga los resultados en las celdas 0B08 a 0B0B

```
MOV R4, [0B00]
MOV R5, [0B01]
CALL promedio
MOV [0B08], R6
MOV R4, [0B02]
MOV R5, [0B03]
CALL promedio
MOV [0B09], R6
MOV R4, [0B04]
MOV R5, [0B05]
CALL promedio
MOV [0B0A], R6
MOV R4, [0B06]
MOV R5, [0B07]
CALL promedio
MOV [0B0B], R6
```

Ejercicio: usar la rutina **promedio** para calcular el promedio de 4 estudiantes de una comisión, cuyas notas están en las celdas 0B00 a 0B07, y ponga los resultados en las celdas 0B08 a 0B0B

```
MOV R4, [0B00]      ←Parámetro
MOV R5, [0B01]      ←Parámetro
CALL promedio
MOV [0B08], R6
MOV R4, [0B02]      ←Parámetro
MOV R5, [0B03]      ←Parámetro
CALL promedio
MOV [0B09], R6
 MOV R4, [0B04]     ←Parámetro
MOV R5, [0B05]      ←Parámetro
CALL promedio
MOV [0B0A], R6
MOV R4, [0B06]      ←Parámetro
MOV R5, [0B07]      ←Parámetro
CALL promedio
MOV [0B0B], R6
```

Ejercicio: documentar la rutina potencia4

```
potencia4   MUL R0,R1
            MUL R0,R1
            MUL R0,R1
            MOV R2,R0
            RET
```

Ejercicio: documentar la rutina potencia4

```
potencia4   MUL R0,R1
            MUL R0,R1
            MUL R0,R1
            MOV R2,R0
            RET
```

Requiere El valor a potenciar en R1

Retorna La potencia $R1^4$ en R2, que está en el rango $[0.,2^{16}]$

Modifica R0

Ejercicio: documentar la rutina descontar:

```
descontar:  MOV R6, R3
            MUL R6, R4
            DIV R6, 0x0064
            SUB R3, R6
            RET
```

Ejercicio: documentar la rutina descontar:

```
descontar:  MOV R6, R3
            MUL R6, R4
            DIV R6, 0x0064
            SUB R3, R6
            RET
```

Requiere El valor original en R3

Retorna El valor con el descuento en R3

Modifica R6

La Pila

¿Qué es una pila?

- La pila es una estructura para almacenar datos

¿Qué es una pila?

- La pila es una estructura para almacenar datos
- Los datos se organizan *apilados*

¿Qué es una pila?

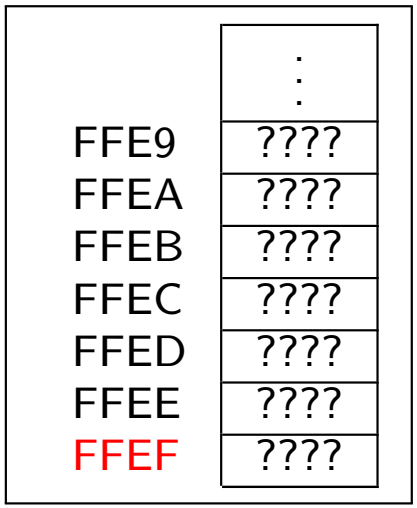
- La pila es una estructura para almacenar datos
- Los datos se organizan *apilados*
- Se utiliza mediante dos operaciones:

  Apilar-push Cuando se escribe en la pila, se lo agrega "sobre" el último agregado

  Desapilar-pop Cuando se lee de la pila, se lo saca del "tope" de la pila

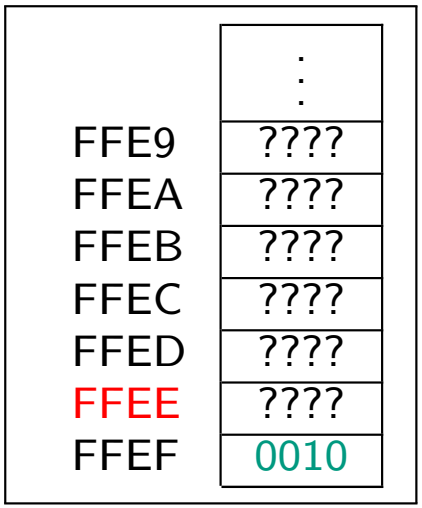
Funcionamiento de la pila

1. Estado original: Tope de pila en FFEF



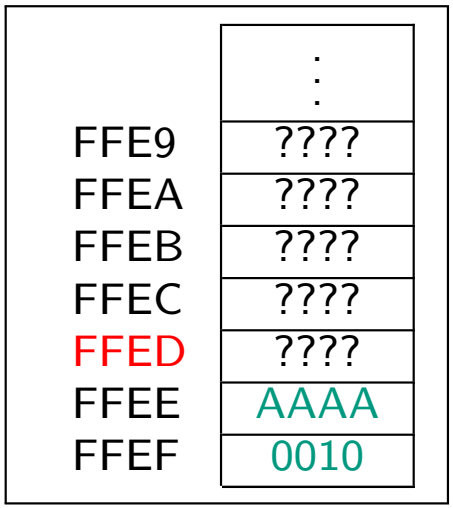
Funcionamiento de la pila

1. Estado original: Tope de pila en FFEF
2. Apilar elemento: 0010. Tope de pila en FFEE



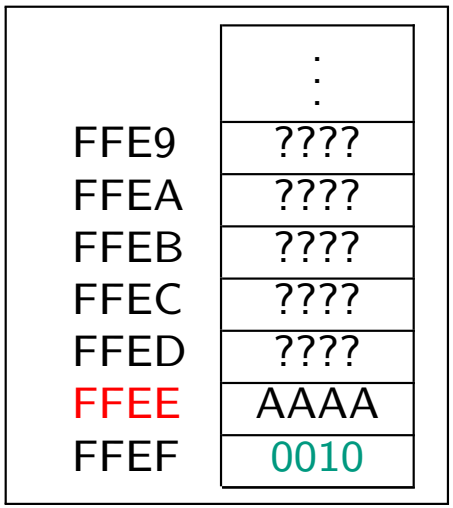
Funcionamiento de la pila

1. Estado original: Tope de pila en FFEF
2. Apilar elemento: 0010. Tope de pila en FFEE
3. Apilar elemento: AAAA. Tope de pila en FFED



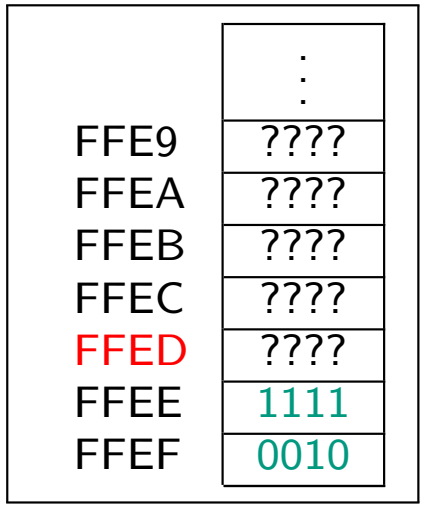
Funcionamiento de la pila

1. Estado original: Tope de pila en FFEF
2. Apilar elemento: 0010. Tope de pila en FFEE
3. Apilar elemento: AAAA. Tope de pila en FFED
4. Desapilar elemento. Tope de pila en FFEE



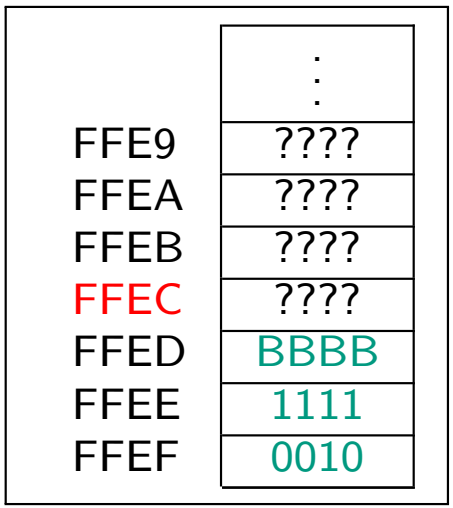
Funcionamiento de la pila

1. Estado original: Tope de pila en FFEF
2. Apilar elemento: 0010. Tope de pila en FFEE
3. Apilar elemento: AAAA. Tope de pila en FFED
4. Desapilar elemento. Tope de pila en FFEE
5. Apilar elemento: 1111. Tope de pila en FFED



Funcionamiento de la pila

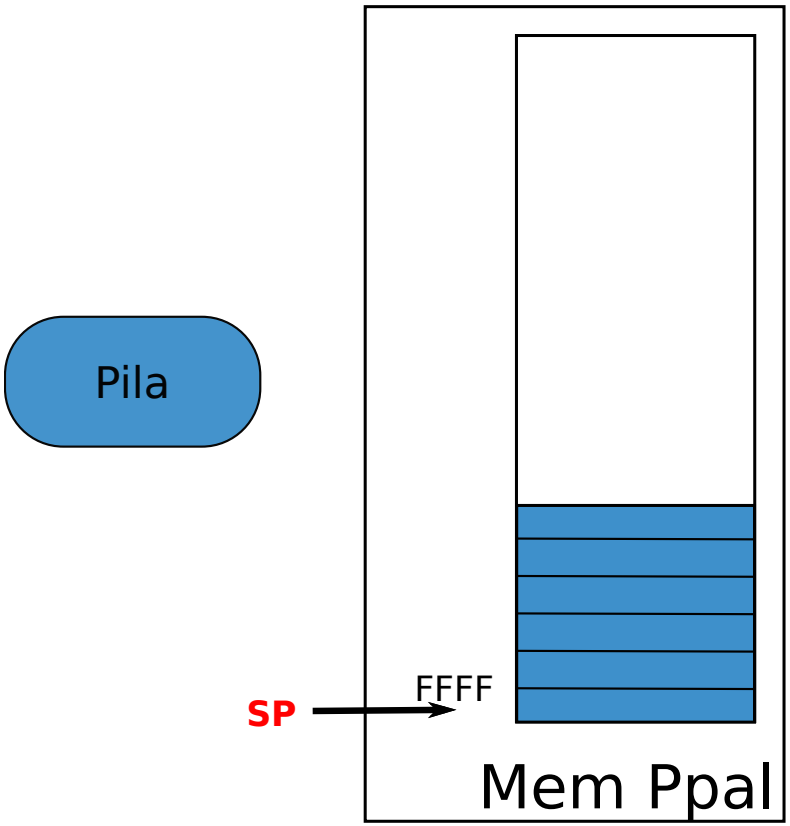
1. Estado original: Tope de pila en FFEF
2. Apilar elemento: 0010. Tope de pila en FFEE
3. Apilar elemento: AAAA. Tope de pila en FFED
4. Desapilar elemento. Tope de pila en FFEE
5. Apilar elemento: 1111. Tope de pila en FFED
6. Apilar elemento: BBBB. Tope de pila en FFEC



Implementación de Pila

- La pila es un sector especial de la memoria
- El seguimiento del tope de pila se lleva mediante un registro especial SP (Stack Pointer)

Implementación de Pila



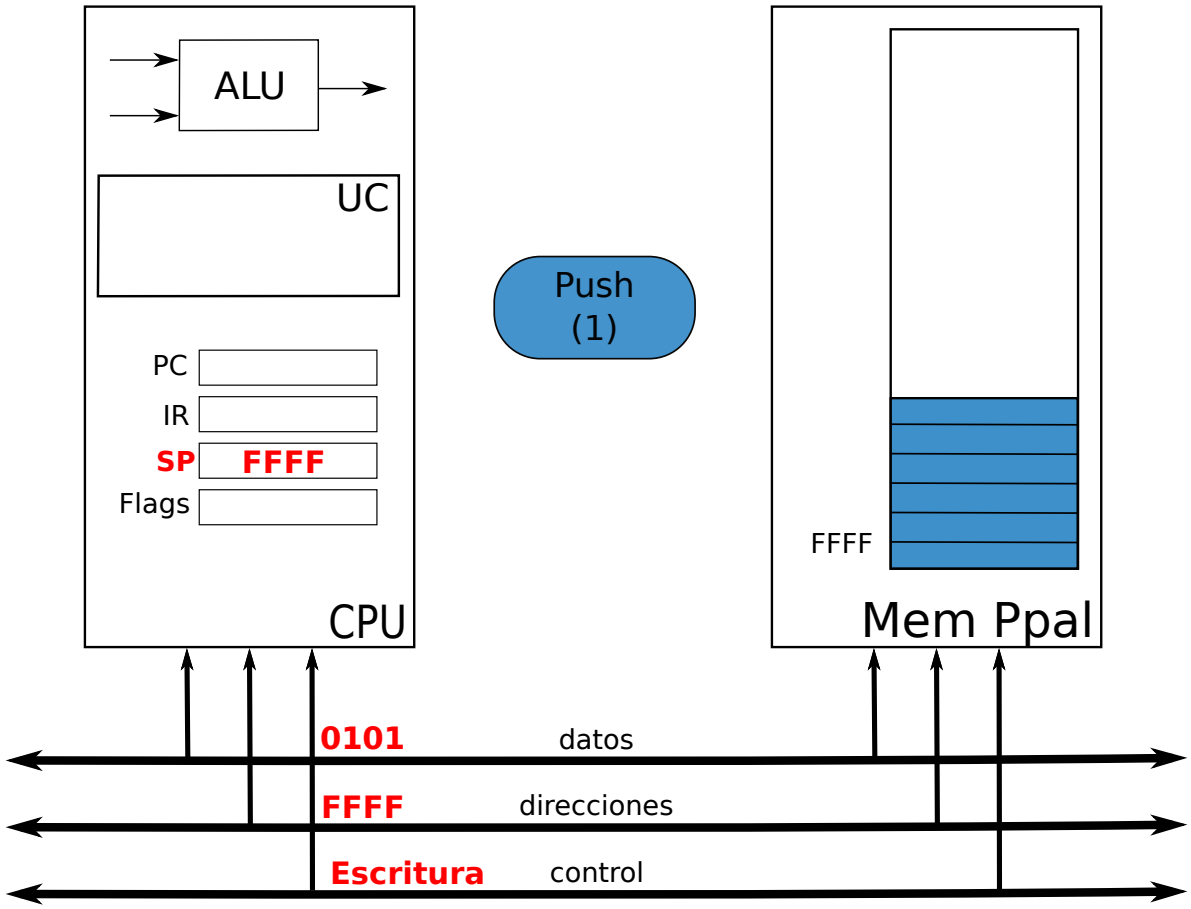
SP (Stack Pointer) contiene la dirección de la primer celda de memoria **disponible** de la pila.

Estructura de Pila: Push

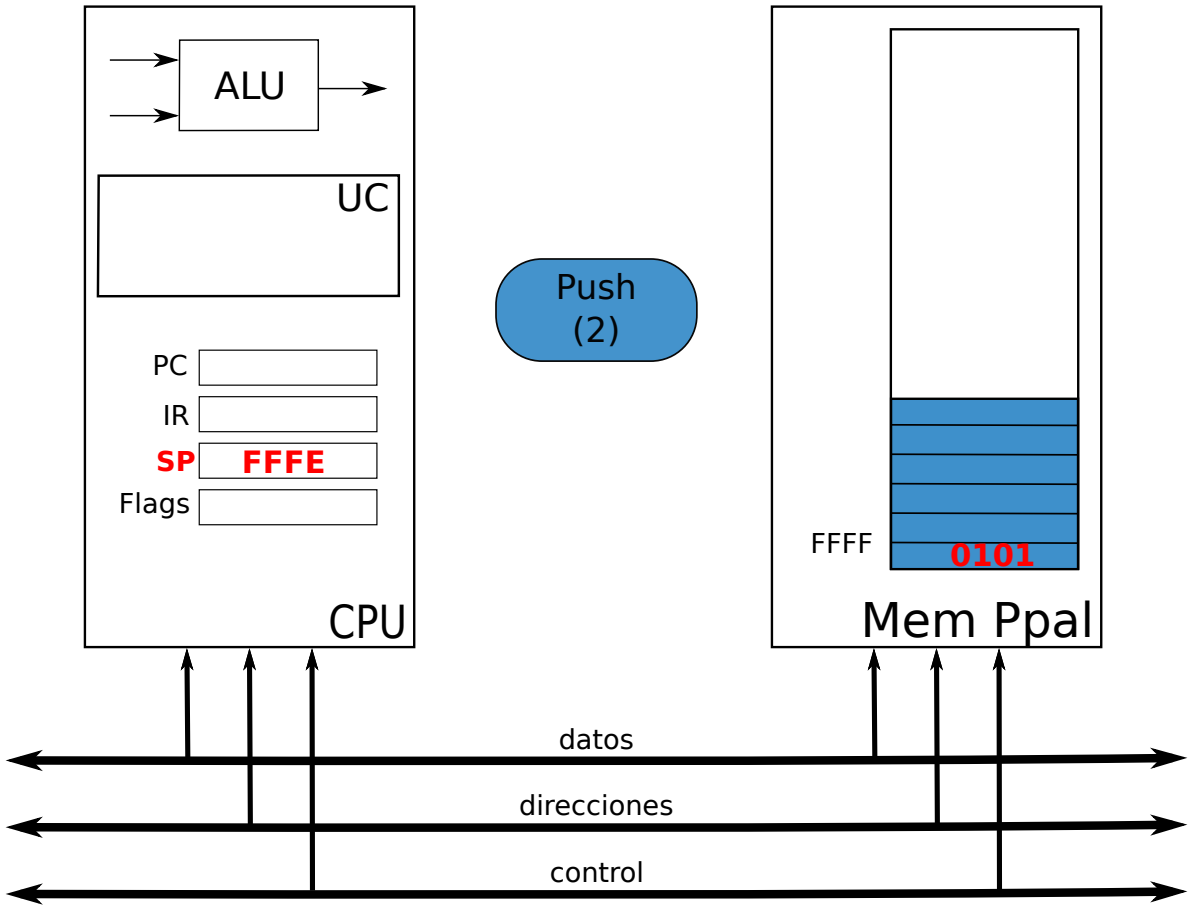
Push

1. Se hace una escritura del dato que está en el bus de datos en la dirección que está en SP
2. Se decrementa SP (así sigue cumpliendo la condición)

Estructura de Pila: Push



Estructura de Pila: Push

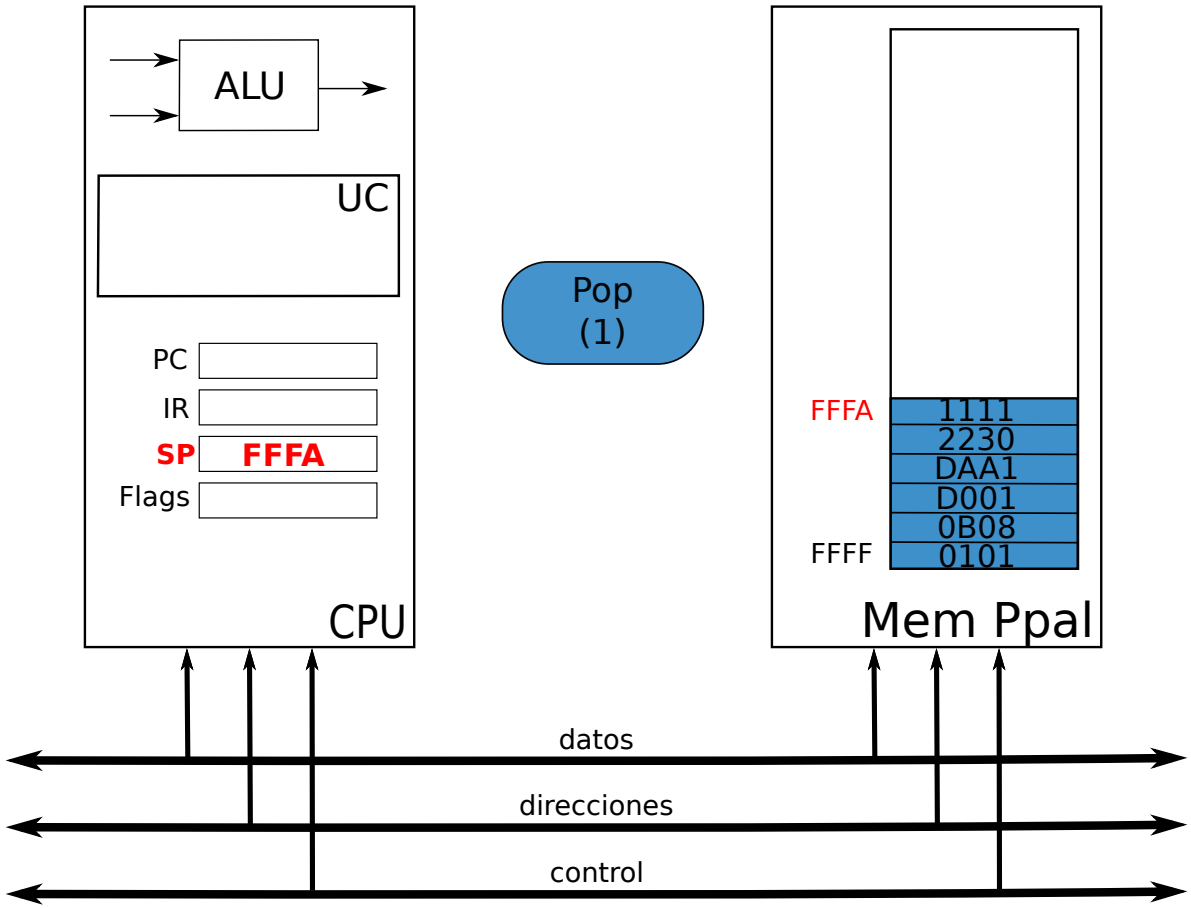


Estructura de Pila: Pop

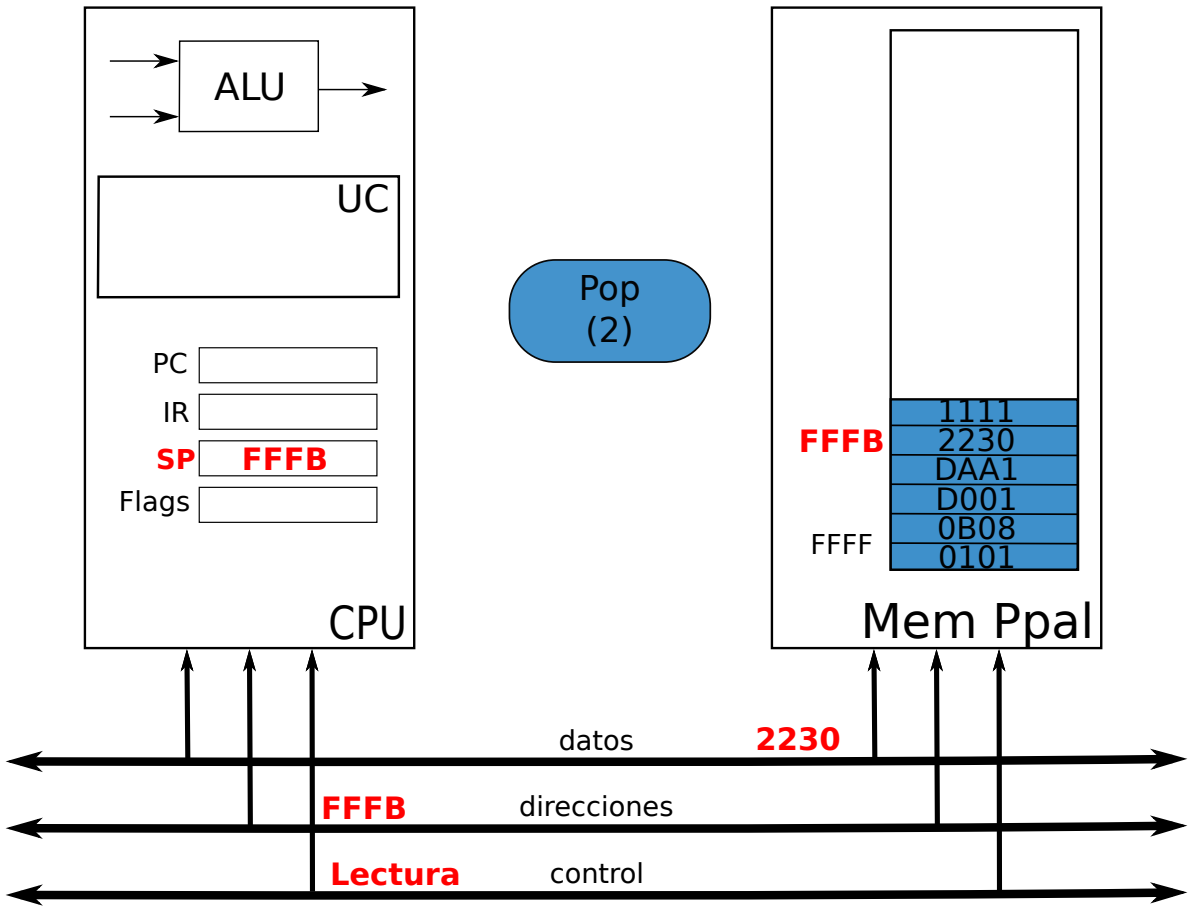
Pop

1. Se incrementa SP (para que haga referencia a un dato dentro de la pila)
2. Se hace una lectura de la dirección que está en SP

Estructura de Pila: Pop



Estructura de Pila: Pop



Estructura de Pila

- El tamaño y la ubicación de la pila está definido por la arquitectura.
- El pop no blanquea el tope de la pila.
- Cuando se hace push se pierde el valor que tenía la celda (por definición de escritura)

¿Cómo funcionan CALL y RET?

¿Cómo funcionan 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 del último llamado

El último llamado es el primero que se restituye

¿Cómo funcionan CALL y RET?

El último llamado es el primero que se restituye

... me resulta conocido...

¿Cómo funcionan CALL y RET?

El último llamado es el primero que se restituye

... me resulta conocido...

¡Se usa una pila!

¿Cómo funcionan CALL y RET?

CALL **Apila** la dirección de la siguiente instrucción

RET **Desapila** la dirección que había quedado pendiente en el programa que llamó.

¿Cómo funcionan CALL y RET?

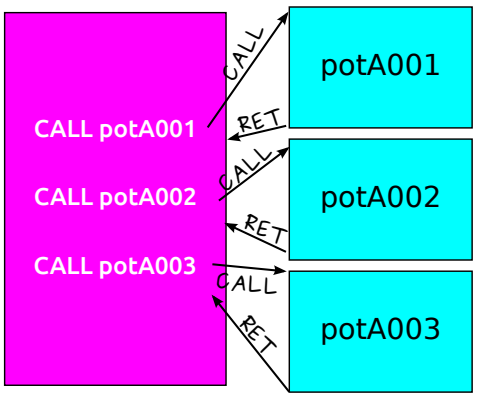
Ajá... ¿Y cómo se sabe la dirección **siguiente**?

¿Cómo funcionan CALL y RET?

Ajá... ¿Y cómo se sabe la dirección **siguiente**?

¡Program Counter!

¿Cómo funcionan CALL y RET?



1. **CALL potA001**: se apila la dirección del **CALL potA002**
2. **RET**: se desapila la dirección del **CALL potA002**
3. **CALL potA002**: se apila la dirección del **CALL potA003**
4. **RET**: se desapila la dirección del **CALL potA003**
5. **CALL potA001**: se apila la dirección de la siguiente instrucción

Ejercicio: mostrar cómo varía la pila

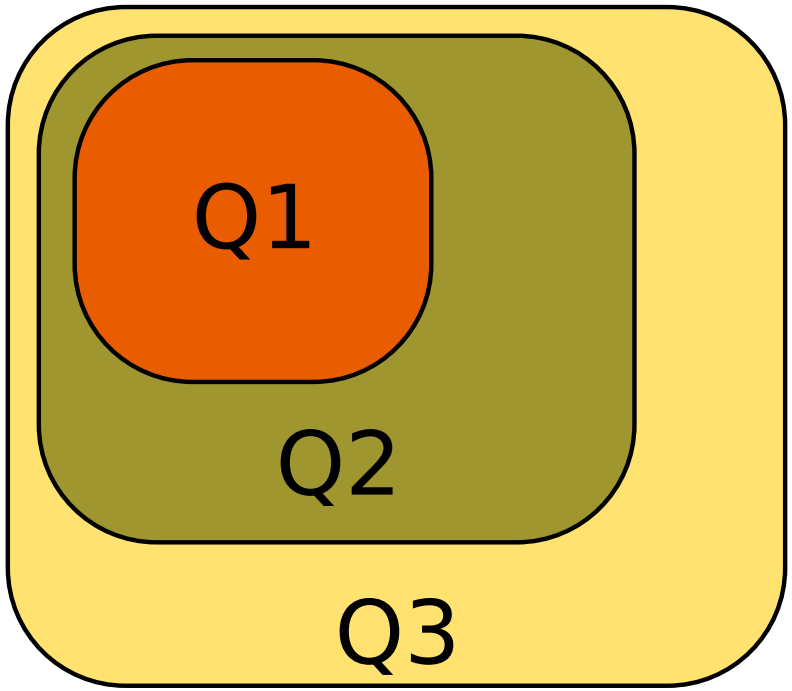
```
rutinaA:    MOV R0, R1
            CALL rutinaB
            RET

rutinaB:    SUB R0, 0x0003
            RET

        CALL rutinaA
```

Arquitecturas **Q**

... tercer acto ...

Arquitectura **Q3**

Arquitectura **Q3**



Arquitectura Q3

- Tiene 8 registros de uso general de 16 bits: R0..R7
- Tiene direcciones de 16 bits
- Tiene registros no visibles al programador:
  - *Program counter* de 16 bits.
  - *Stack Pointer* de 16 bits. Comienza en la dirección FFEF.
- permite 3 modos de direccionamiento:
  - modo registro: el valor buscado está en un registro
  - modo inmediato: el valor buscado está codificado dentro de la instrucción
  - modo directo: el valor buscado está contenido en una celda de memoria

Arquitectura **Q3**: formato de instrucciones

- Instrucciones de 2 operandos (MUL,MOV,ADD,SUB,CMP,DIV,AND,OR)

Cod_Op (4b)	Modo Destino (6b)	Modo Origen (6b)	Operando Destino (16b)	Operando Origen (16b)

- Instrucciones con un operando Origen: CALL

Cod_Op (4b)	Relleno (000000)	Modo Origen (6b)	Operando Origen (16b)

- Instrucciones sin operandos: RET

Cod_Op (4b)	Relleno (000000000000)

Arquitectura **Q3**: Instrucciones Aritméticas y lógicas

Cod_Op (4b)	Modo Destino (6b)	Modo Origen (6b)	Operando Destino (16b)	Operando Origen (16b)

Operación	**CodOp**
MUL	0000
MOV	0001
ADD	0010
SUB	0011
CMP	0110
DIV	0111
AND	0100
OR	0101

Arquitectura **Q3**: Instrucciones con un operando Origen

Cod_Op (4b)	Relleno (000000)	Modo Origen (6b)	Operando Origen (16b)

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

Arquitectura **Q3**: Instrucciones sin operandos

Tipo 5: Instrucciones sin operandos

Cod_Op (4b)	Relleno (000000000000)

Operación	**CodOp**	**Efecto**
RET	1100	PC $\leftarrow$ [SP+1]; SP $\leftarrow$ SP + 1