Soluciones-Enero-2024.pdf

MeGustaElDinero

Arquitectura de Computadores

2º Grado en Ingeniería Informática

Escuela Politécnica Superior de Córdoba
Universidad de Córdoba

Accede al documento original

Examen final A.C.

16/01/24

-Enunciados:

TEORÍA

1. *La organización de la pila explicada en la clase seguía un protocolo para escribir y recuperar datos en la pila de modo que SP siempre apuntaba a la última posición (de la pila). Imagínese que se quiere cambiar este protocolo para que SP pase a apuntar a la primera posición libre (de la pila). Por tanto:* (**1 punto**)

a) Escribir el nuevo protocolo para guardar algo en el tope de la pila.

b) Escribir el nuevo protocolo para recuperar algo del tope de la pila.

c) Si antes se inicializa SP con 00 *(hexadecimal) ¿Con cuál dirección debería inicializarse ahora?*

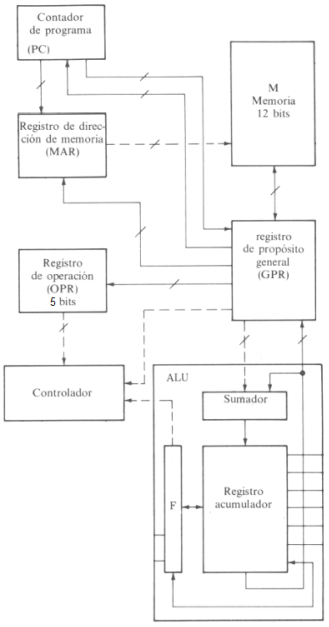
2. *Los algoritmos de multiplicación de números de complemento a* 2 *tiene en común un problema que no tiene el algoritmo de Booth ¿Qué problema es?* (**1 punto**)

3. *Supongamos que en el siguiente esquema de la computadora mejorada se modifica OPR pasando de* 4 *a* 5 *bits. Indicar las siguientes cuestiones, justificando brevemente su respuesta:* (**1 punto**)

a) En el formato instrucción ¿Cuántos bits se destinan a instrucción y cuantos se destinan a campo operando?

b) Tamaño de Memoria Principal en nº de palabras por ancho de palabra.

c) Señalar en el esquema, junto al nombre de cada registro, el número de bits que almacena.



PROBLEMAS

4. *Desarrollar un programa mediante instrucciones MIPS que calcule los* $(n+1)$ *primeros elementos de la sucesión de Fibonacci y guarde los correspondientes elementos en el vector A, sabiendo que el valor n se encuentra en el registro* $\$\boldsymbol{s0}$ *y la dirección base del vector A se encuentra en el registro* $\$\boldsymbol{s1}$.

La sucesión de Fibonacci viene definida por:

$$f_i = f_{i-1} + f_{i-2}, \ \ para\ i = 2, \dots, n, siendo\ f_0 = 0\ y\ f_1 = 1$$

Y se pide más concretamente, que el programa haga lo siguiente:

$$A[0] = f_0\ ;\ A[1] = f_1\ ; \dots ;\ A[n] = f_n$$

Considerese $n \geq 2$. (**2 puntos**)

5. *Considerando una Memoria Principal de* $512K \times 12$, *y una Memoria Caché asociativa por conjunto de* 2 *vías, que utiliza* 2 *bloques de* 8 *palabras, pudiendo almacenar* $4K$ *palabras de Memoria Principal.* (**3 puntos**)

a) *Determinar la organización de Memoria Caché.*

b) *Determinar el tamaño de Memoria Caché.*

c) *Teniendo en cuenta que en Memoria Principal se encuentran* 2 *vectores de* 12 *bits por elemento almacenados en los siguientes esquemas*:

Vec. A: 6AF28	8	13	21	0	77	3	98	52	1	43
Vec. B: 0EF28	122	40	28	22	17	41	12	68	15	80

Representar el contenido tras 6 *iteraciones del bucle*:

```
1: int sp;
2:
3: sp = 0;
4: for (int i=0; i<n; ++i)
5:     sp += A[ i]*B[ i];
```

6. *Considerando la estructura de computadora mejorada mostrada en la figura de la primera página, implemente la instrucción* "RIZ m " *que calcula el número el número de ceros que se encuentran a la derecha del primer* 1 *menos significativo, y lo envía a la posición de memoria m. En el caso de que todo sea* 0, *se devolverá a memoria una palabra con todos los bits a* 1. (12 *bits a* 1). (**2 puntos**) *Hágalo con*:

a) *Control Microprogramado* (*con ensamble LCB*).

b) *Control cableado.*

Incluya el ciclo de búsqueda y los terminales de control y bits de estado que crea necesarios para implementar la instrucción. Se puede incluir registros de propósito especial como QR.

-Soluciones:

Ejercicio 1:

a) Si la pila apunta a la primera posición libre, significa que el nuevo protocolo para escribir en la pila es:

1. Primero escribe el dato en la posición de memoria que contiene el registro SP.

2. Luego decrementa SP en uno para que apunte a la siguiente posición vacía.

b) Al igual que en el caso anterior, se intercambian los pasos del protocolo de extracción de palabra de la pila respecto al sistema que apunta al último dato apilado:

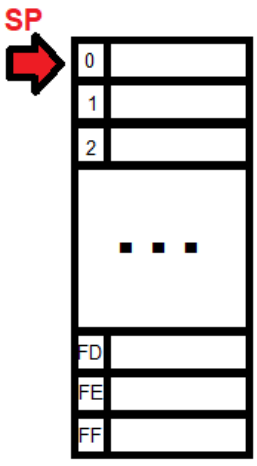
1. Primero se incrementa en uno el registro SP.

2. Luego saca el dato de la pila y lo almacena en GPR.

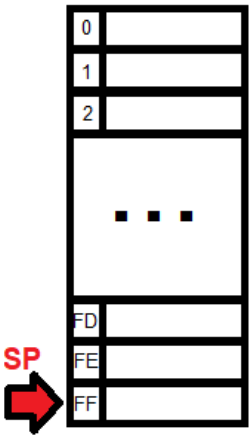
c) Si la primera posición de la memoria es 00 en hexadecimal, según este nuevo protocolo la pila estaría apuntando a la posición FF en hexadecimal. (Considerando que el tamaño de memoria sea de 2^8).

Contenido registro SP cuando está vacía la pila

Formato tope de pila

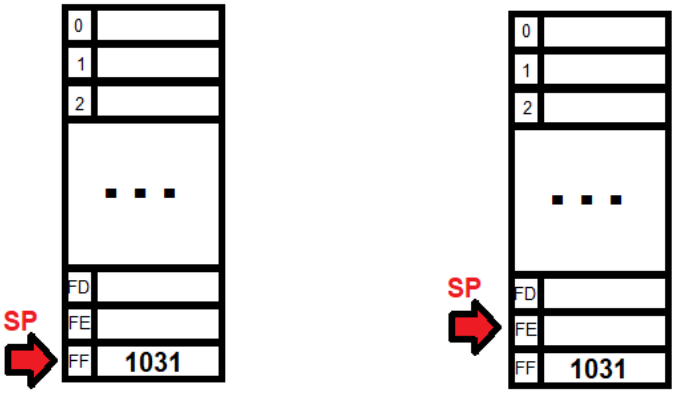


Formato última pos. libre



Contenido registro SP cuando hay un elemento en la pila

Formato tope de pila **Formato última pos. libre**



Ejercicio 2:

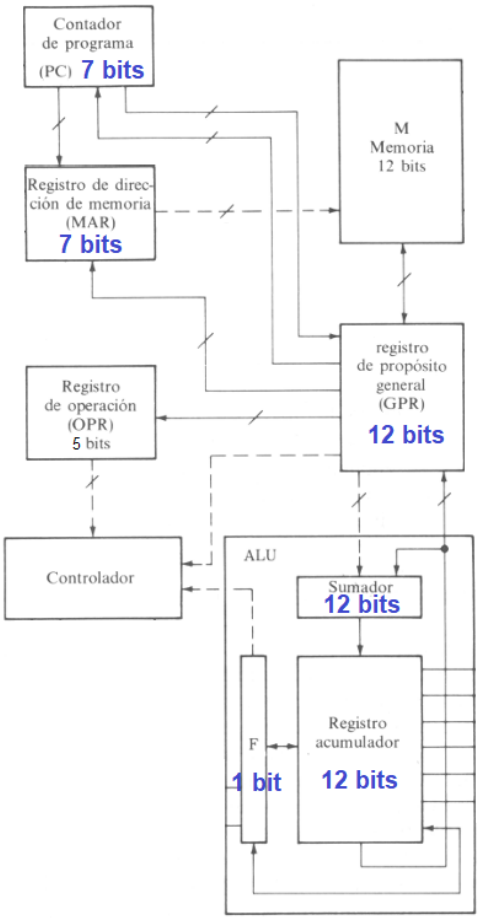
Los inconvenientes del algoritmo complemento a 2 en la multiplicación respecto al algoritmo de Booth es que no es necesaria la comprobación de signos en el algoritmo de Booth antes de la multiplicación ya que esta es independiente del resultado final que genera.

Ejercicio 3:

a) Si tenemos una computadora mejorada en la que se nos dice que el registro OPR (el registro que se encarga de almacenar la instrucción de una palabra), tiene una capacidad de 5 bits, significa que **la cantidad de palabra que se le destina a la instrucción son 5**. Si el ancho de palabra de memoria es de 12 bits, **la cantidad de bits que se destinan a la dirección de memoria** (los bits que se destinan al registro MAR) **son:** 12-5= **7 bits**.

b) Como tenemos 7 bits para el registro MAR, que ese tamaño significa que únicamente puede alcanzar a direccionar a direcciones de memoria cuya posición esté determinada por 7 bits. Por lo que la capacidad de memoria son **2^7 palabras.** Y además sabiendo que el ancho de palabra de memoria es de 12 bits, la capacidad de MP es: **2^7 x12= 128x12.**

c)

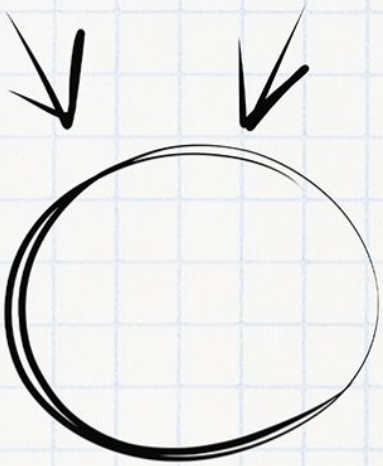


Imagínate aprobando el examen

Necesitas tiempo y concentración

Planes	PLAN TURBO	PLAN PRO	PLAN PRO+
Descargas sin publi al mes	10	40	80
Elimina el video entre descargas	✓	✓	✓
Descarga carpetas	✗	✓	✓
Descarga archivos grandes	✗	✓	✓
Visualiza apuntes online sin publi	✗	✓	✓
Elimina toda la publi web	✗	✗	✓
Precios Anual	0,99 € / mes	3,99 € / mes	7,99 € / mes

Ahora que puedes conseguirlo, ¿Qué nota vas a sacar?



WUOLAH

Ejercicio 4:

Para poder entender el código escrito en MIPS más fácilmente también el siguiente código en C (no es necesario para el examen):

```c
int main(){
    int N, a= 0, b= 1;
    int A[N];
    A[0]= a;
    A[1]= b;
    for(int i=2; i<N; i++){
        A[i]= b+a;
        a= b;
        b= A[i];
    }
    return 0;
}
```

El código en MIPS es:

```
    # $t0= a;  $t1= b;  $t2= i, $s0= N, $s1= dir A[0];

        add $t0, $zero, $zero     # t0= 0+0;  t0= 0
        sw $t0, 0($s1)            # A[0]= t0;  A[0]= t0
        addi $t1, $t0, 1          # t1= t0+1;  t1= 0+1;  t1= 1
        sw $t1, 4($s1)            # A[4/4]= t1;  A[1]= 1
        addi $t2, $t1, 1          # t2= t1+1;  t2= 1+1;  t2= 2

Loop:   sll $t3, $t2, 2           # t3= t2*4
        add $t3, $t3, $s1         # t3= t3+ dir A[0];  t3= dir A[t2];
        add $t4, $t0, $t1         # t4= t0+t1;  t4= a+b
        sw $t4, 0($t3)            # A[t2]= t4;  A[i]= a+b
        add $t0, $t1, $zero       # t0= t1+0;  a= b+0;  a= b;
        lw $t1, 0($t3)            # t1= A[t3];  b= A[i]
        addi $t2, $t2, 1          # t2= t2+1;  i= i+1;  i++
        slt $t5, $t2, $s0         # t5= 1 si t2<s0; t5=1 si i<N
        bne $t5, $zero, Loop      # si t5= 1 vuelve a Loop, si t5= 0 termina
```

Ejercicio 5:

Según dice el problema tenemos:
M.P.: 512K palabras= **2^{19} palabras.**

M.C.: 4K palabras= **2^{12} palabras.**

La memoria caché es asociativa por conjunto de 2 vías y tiene 2 bloques de 8 palabras. El número de palabras por línea es igual a la suma de las palabras en los bloques de cada vía: **8x2= 2^3 x 2^1.**

a)

El formato de direccionado tiene una longitud de 19 bits debido a:

$$Bits\ de\ formato\ de\ dirección = \log_2(Cant.\ palabras\ MP) = \log_2(2^{19}) = \mathbf{19\ bits}$$

Para conocer el número de líneas de la memoria caché se aplica la siguiente fórmula:

$$Cantidad\ de\ palabras\ M.C. = nº\ de\ líneas \times nº\ de\ palabras/línea$$

$$2^{12} = nº\ de\ líneas \times (2^3 \times 2^1); \quad nº\ de\ líneas = \frac{2^{12}}{2^4} = \mathbf{2^8\ líneas}$$

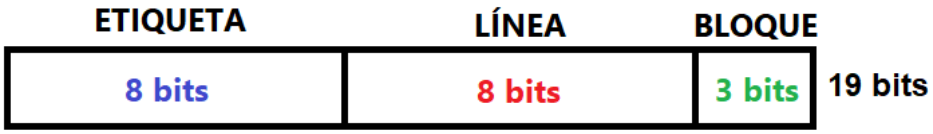
Para obtener la cantidad de bits de cada campo:

$$Bits\ de\ \mathbf{bloque} = \log_2(Cant.\ palabras/\ bloque) = \log_2(2^3) = \mathbf{3\ bits}$$

$$Bits\ de\ \mathbf{línea} = \log_2(Cant.\ líneas\ MC) = \log_2(2^8) = \mathbf{8\ bits}$$

$$Bits\ de\ \mathbf{etiqueta} = Bits\ totales - (Bits\ línea + Bits\ de\ bloque) = 19 - (8+3) = \mathbf{8\ bits}$$

Formato de dirección de palabra



b)

Para calcular el tamaño de memoria caché aplicamos la siguiente fórmula:

$$Capacidad\ Caché = nº\ líneas\ \times\ nº\ bits/línea$$

Los bits de una línea están compuestos por, los bits de la etiqueta del formato de dirección de palabra **(8)**, más la cantidad de bits por palabra la cual aparece en la información de la Memoria Principal **(12)**, por la cantidad de palabras por bloque **(8),** y más el bit de validación **(1)**. Estos bits se multiplican en memorias caché asociativas por conjunto por el número de bloques/vías que contiene la memoria **(en este caso 2).** La solución sería:

$$Capacidad\ Caché = \ 2^8 \times [(8 + [12\ \times 8] + 1) \times 2]$$

$$Capacidad\ Caché = \ 2^8 \times [210] = \mathbf{53,76\ Kbits}$$

c)

Según el formato de dirección de palabra, las direcciones de los vectores A y B, se dividirían de la siguiente manera:

	Etiqueta	Línea	Bloque
$Vector\ A\ (6AF28)$:	110 1010	1111 0010	1000
$Vector\ B\ (0EF28)$:	000 1110	1111 0010	1000

Según vemos, tenemos que ambas direcciones comparten los mismos bits de línea por lo que, al ser una memoria caché asociativa por conjunto, en la primera vía se almacenarán los elementos del vector A, y en la segunda vía se almacenarán los elementos del vector B. La disposición tras 6 iteraciones será de la siguiente manera:

	VÍA 1									VÍA 2								
Línea	Etiqueta	P_0	P_1	P_2	P_3	P_4	P_5	P_6	P_7	B_V	Etiqueta	P_0	P_1	P_2	P_3	P_4	P_5	P_6	P_7	B_V
000 0000 0	000 0000 0									0	000 0000 0									0
...	...										...									
111 0010 1	110 1010 1	8	13	21	0	77	3	98	52	1	000 1110 1	122	40	28	22	17	41	12	68	1
...	...										...									
111 1111 1	111 1111 1									0	111 1111 1									0

Ejercicio 6:

(Explicaciones al final de la tabla de control de cableado)

Control micro programado:

Dirección CROM	Micro operaciones	LCB			Dirección bifurcación
		B_2	B_1	B_0	
ADDR(FETCH)+0	PC-> MAR	0	0	1	-
ADDR(FETCH)+1	PC+1-> PC; M->GPR	0	0	1	-
ADDR(FETCH)+2	GPR(OP)-> OPR; GPR(AD)-> MAR	0	1	1	-
ADDR(RIZ)+0	12-> SC; QR-> M	0	0	1	-
ADDR(RIZ)+1	0-> QR; SP-1-> SP	0	0	1	-
ADDR(RIZ)+2	SP-> MAR	0	0	1	-
ADDR(RIZ)+3	GPR-> M	0	0	1	-
ADDR(RIZ)+4	QR->GPR	0	0	1	-
ADDR(RIZ)+5	ROR F Acc; SC-1-> SC	0	0	1	-
ADDR(RIZ)+6	GPR+1-> GPR	1	0	0	**ADDR(RIZ)+9**
ADDR(RIZ)+7	$\overline{Acc} \rightarrow Acc$	1	0	1	**ADDR(RIZ)+5**
ADDR(RIZ)+8	Acc-> GPR	0	0	1	**-**
ADDR(RIZ)+9	GPR-> QR	0	0	1	-
ADDR(RIZ)+A	M-> GPR	0	0	1	-
ADDR(RIZ)+B	GPR(AD)-> MAR; SP+1->SP	0	0	1	-
ADDR(RIZ)+C	M-> GPR	0	0	1	-
ADDR(RIZ)+D	QR-> M	0	0	1	-
ADDR(RIZ)+E	GPR-> QR	0	1	0	**ADDR(FETCH)+0**

Tabla de control de bifurcación:

B_2	B_1	B_0	F	Z_{SC}	...	I	B	R	E
0	0	1	X	X		1	0	0	1
0	1	0	X	X		0	1	0	1
0	1	1	X	X		0	0	1	1
1	0	0	0	X	...	1	0	0	1
1	0	0	1	X		0	1	0	0
1	0	1	X	0		0	1	0	0
1	0	1	X	1		1	0	0	1

Control cableado

Condiciones lógicas	Micro operaciones	Siguiente
t_0	PC-> MAR	SR+1-> SR
t_1	PC+1-> PC; M->GPR	SR+1-> SR
t_2	GPR(OP)-> OPR; GPR(AD)-> MAR	SR+1-> SR
$i_1 \cdot t_3$	12-> SC; QR-> M	SR+1-> SR
$i_1 \cdot t_4$	0-> QR; SP-1-> SP	SR+1-> SR
$i_1 \cdot t_5$	SP-> MAR	SR+1-> SR
$i_1 \cdot t_6$	GPR-> M	SR+1-> SR
$i_1 \cdot t_7$	QR->GPR	SR+1-> SR
$i_1 \cdot t_8$	ROR F Acc; SC-1-> SC	SR+1-> SR
$i_1 \cdot t_9 \cdot \bar{F}$	GPR+1-> GPR	SR+1-> SR
$i_1 \cdot t_9 \cdot F$	-	12->SR
$i_1 \cdot t_{10} \cdot \overline{Z_{sc}}$	-	8-> SR
$i_1 \cdot t_{10} \cdot Z_{sc}$	$\overline{Acc} \rightarrow Acc$	SR+1-> SR
$i_1 \cdot t_{11}$	Acc-> GPR	SR+1-> SR
$i_1 \cdot t_{12}$	GPR-> QR	SR+1-> SR
$i_1 \cdot t_{13}$	M-> GPR	SR+1-> SR
$i_1 \cdot t_{14}$	GPR(AD)-> MAR; SP+1->SP	SR+1-> SR
$i_1 \cdot t_{15}$	M-> GPR	SR+1-> SR
$i_1 \cdot t_{16}$	QR-> M	SR+1-> SR
$i_1 \cdot t_{17}$	GPR-> QR	0-> SR

Explicación (NO NECESARIO PARA EXÁMEN)

FETCH/ciclo de búsqueda:

Lo primero que hace esta instrucción es enviar a MAR la dirección donde se encuentra la siguiente instrucción a ejecutar **(PC->MAR)**. Ahora que la máquina está apuntando a la dirección de memoria donde se encuentra la siguiente instrucción, descargamos el dato de memoria **(M->GPR)** e incrementamos el valor de PC para que apunte a la siguiente instrucción a ejecutar **(PC+1->PC).** Ahora que tenemos la instrucción en GPR, extraemos la parte de operación de la palabra **(GPR(OP)->OPR)**, y la dirección a la que apunta la instrucción **(GPR(AD)->MAR)** para enviarla a sus respectivos registros para que la computadora los tome en cuenta.

En micro programado es necesario incluir en el LCB la combinación que tiene **R** o **Reload** a **1**, y en las instrucciones anteriores **I** o **Incremento** a **1**. En todas las instrucciones de FETCH es necesario valorar **E** o **Enable** a **1** para que pueda realizar las micro operaciones citadas anteriormente.

En control cableado es necesario escribir **SR+1-> SR** en todas las micro operaciones para que el registro SR pueda incrementar y avanzar.

RIZ m:

Comenzamos ingresando al registro contador SR el valor del ancho de palabra de memoria principal **(12-> SC)**, y en el mismo espacio enviamos el dato que pueda estar en el registro QR a la dirección de memoria que actualmente contiene MAR (en este caso es la dirección de retorno de la instrucción RIZ m como se puede ver en las imágenes inferiores) con **(QR->M).**

Vaciamos QR **(0->QR)** y decrementamos el registro de la pila para ingresar más tarde un dato **(SP-1->SP)**.

El dato decrementado del registro de la pila, lo enviamos a MAR para que esté apuntando a esa dirección de memoria **(SP->MAR).**

Una vez la computadora esté apuntando al nuevo tope de pila, sacamos el dato de la operación RIZ m, **que contiene la dirección donde tenemos que retornar el valor final (m)**, a la memoria **(GPR->M).**

Ahora para poder contar de manera correcta el número de ceros, debemos de ingresar el valor 0 en el GPR e ir incrementándolo según veamos viendo un cero, por lo que para iniciar GPR a 0 debemos de pasar el valor vaciado de QR a GPR **(QR->GPR)**.

Ya que podemos contar de manera correcta el número de ceros del acumulador antes del primer 1, comenzamos con el bucle decrementando el contador **(SC-1->SC)** y pasando el bit menos significativo del acumulador al registro F **(ROR F Acc).**

Según el valor de F, haremos cosas distintas. Si F=0, incrementaremos el registro que cuenta el número de ceros antes del primer 1 **(GPR+1->GPR)** y **incrementamos** la dirección de CROM, o el registro SR. Si F=1 significa que ya debe de parar el bucle, por lo que no hace nada (en micro programado **E** o **Enable** a **0**), y bifurca fuera del bucle. Para ello creamos una nueva combinación de LCB que en este caso es **100** que **depende del valor de F**.

Si F=0, significa que debe de comprobar si el contador a llegado a 0 (por si ya ha comprobado todos los bits del Acc). Para ello creamos una nueva combinación de LCB que en este caso es **101** que **depende del valor del bit de estado Z_{sc}** el cual es 1 si el valor del registro SC es igual a 0 (es decir, si ha recorrido todos los bits del Acc puesto que lo habíamos inicializado al ancho de palabra).

Si el contador NO ha llegado a 0, vuelve al bucle de manera normal a seguir comprobando bits.

Si el contador SÍ ha llegado a 0, significa que debemos de devolver una palabra de 12 bits donde todos sean igual a 1, por lo que para ello complementamos Acc **(Acc'-> Acc),** (ya que, si ha llegado a 0 el contador significa que no ha encontrado ningún 1 en la palabra, y esta está formada por bits igual a 0, y al hacer complemento de ello una palabra de 12 bits iguales a 1). Habrá que pasar el valor complementado al GPR antes de unificar las dos condiciones **(Acc-> GPR)**.

Una vez tenemos en GPR el valor a devolver a la posición de memoria indicada por la instrucción, debemos de hacer que MAR contenga la dirección de retorno. Como lo tenemos almacenado en el tope de pila, tenemos que primero pasar el valor de retorno a un registro temporal **(GPR-> QR).**

Luego descargar la dirección de retorno, de la pila **(M-> GPR).**

Sacar parte de la dirección de la palabra para guardarla en MAR **(GPR(AD)-> MAR)** y devuelvo el registro de pila a su origen **(SP+1-> SP).**

Una vez la computadora apunta a la dirección de retorno, primero debemos descargar el dato de QR previo a esta instrucción que habíamos dejado antes **(M-> GPR).**

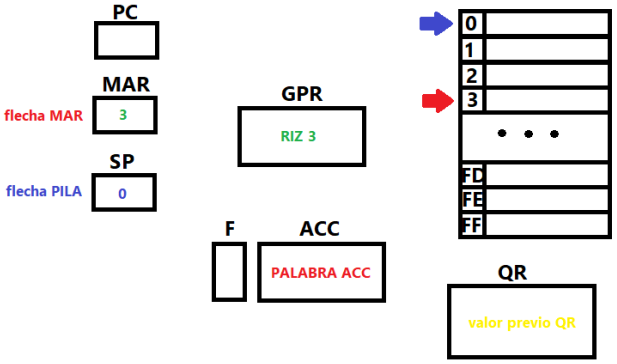
Dejamos en la dirección el valor a retornar **(QR-> M).**

Y por último devolvemos el dato de QR previo a la instrucción a donde estaba **(GPR-> QR).** En esta micro operación debemos de bifurcar a la instrucción FETCH, donde en micro programado sería **ADDR(FETCH)+0** y en control de cableado sería **0-> SR**.

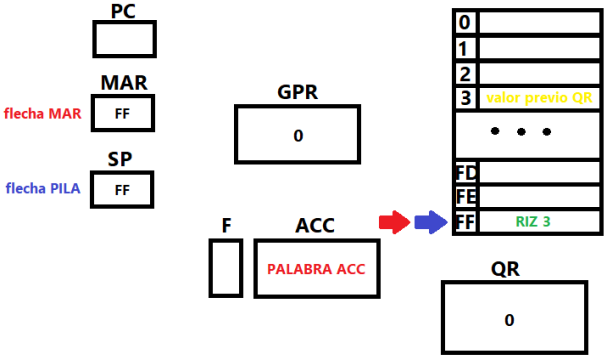
Ejemplo gráfico (NO NECESARIO PARA EXÁMEN):

Considerando que la máquina según el ciclo de búsqueda ingresa la instrucción RIZ 3:

Estado inicial de máquina antes de la instrucción

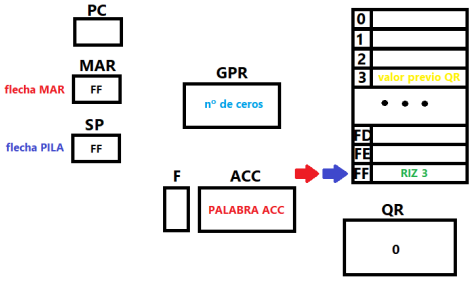


Estado antes de recorrer los bits del acumulador

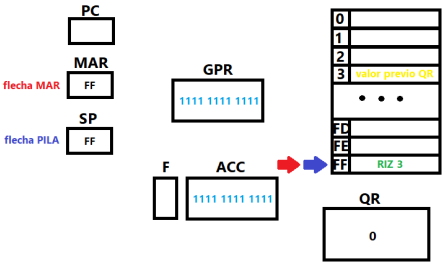


Estado justo después de terminar de iterar

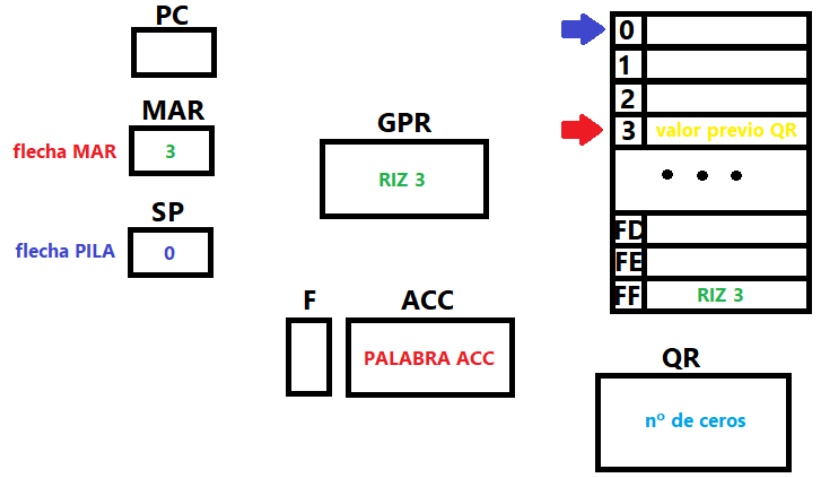
Caso donde hay un 1 en la palabra:



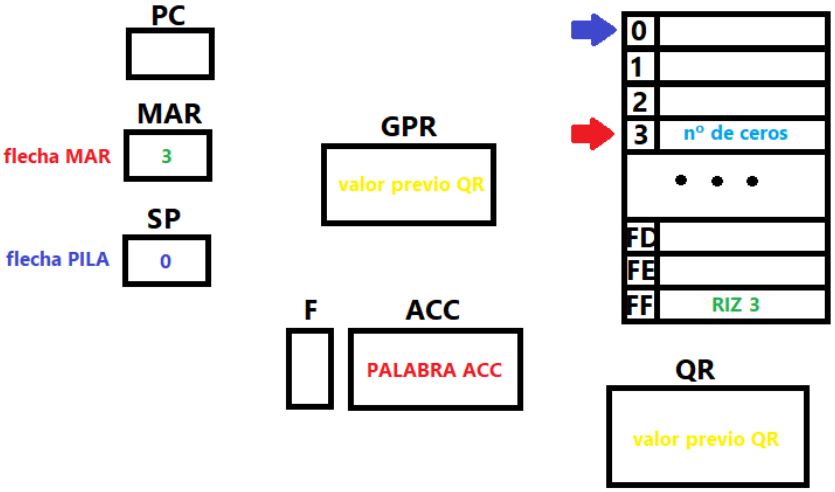
Caso donde todo es 0:



Estado justo antes de ingresar valor a la dirección de retorno



Estado de máquina tras finalizar la instrucción



Hay que considerar que el dato de la pila no es general, ya que depende de la cantidad de elementos que tenga la pila. En este caso se considera el ejemplo de una pila vacía. Al no determinar el tamaño de memoria principal, se considera que la última posición es FF (tamaño de 2^8x12).