Guía de ejercicios # 10: Iteraciones, arreglos y recorridos

Organización de Computadoras 2017

UNQ

Arquitectura Q6

Características

- Tiene 8 registros de uso general de 16 bits: R0..R7.
- La memoria utiliza direcciones de 16 bits.
- Contador de programa (*Program counter*) de 16 bits.
- *Stack Pointer* de 16 bits. Comienza en la dirección `0xFFEF`.
- Flags: Z, N, C, V (Zero, Negative, Carry, oVerflow). Instrucciones que alteran Z y N: ADD, SUB, CMP, DIV, MUL, AND, OR, NOT. Las 3 primeras además calculan C y V.

Instrucciones de dos operandos

Formato de Instrucción

CodOp (4b)	Modo Destino (6b)	Modo Origen (6b)	Destino (16b)	Origen (16b)

Tabla de instrucciones

Operación	Cod Op	Efecto
MUL	0000	Dest ← Dest * Origen
MOV	0001	Dest ← Origen
ADD	0010	Dest ← Dest + Origen
SUB	0011	Dest ← Dest - Origen
AND	0100	Dest ← Dest ∧ Origen
OR	0101	Dest ← Dest ∨ Origen
CMP	0110	Dest - Origen
DIV	0111	Dest ← Dest % Origen

Instrucciones de un operando origen

Formato de Instrucción

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

Tabla de instrucciones

Operación	Cod Op	Efecto
CALL	1011	[SP] ← PC; SP ← SP - 1; PC ← Origen
JMP	1010	PC ← Origen

Instrucciones de un operando destino

Formato de Instrucción

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

Tabla de instrucciones

Operación	Cod Op	Efecto
NOT	1001	Dest ← ¬Dest

Instrucciones sin operandos

Formato de Instrucción

CodOp (4b)	Relleno (000000000000)

Tabla de instrucciones

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

Saltos condicionales

Formato de Instrucción

Cod_Op (8)	Desplazamiento(8)

Los primeros cuatro bits del campo `Cod_Op` es la cadena 1111_2. Si **al evaluar la condición de salto** el resultado es 1, se le suma al PC el valor del desplazamiento, representado en $CA2(8)$. En caso contrario la instrucción no hace nada.

Codop	Op.	Descripción	Condición de Salto
0001	JE	Igual / Cero	Z
1001	JNE	No igual	not Z
0010	JLE	Menor o igual	Z or (N xor V)
1010	JG	Mayor	not (Z or (N xor V))
0011	JL	Menor	N xor V
1011	JGE	Mayor o igual	not (N xor V)
0100	JLEU	Menor o igual sin signo	C or Z
1100	JGU	Mayor sin signo	not (C or Z)
0101	JCS	Carry / Menor sin signo	C
0110	JNEG	Negativo	N
0111	JVS	Overflow	V

Modos de direccionamiento

Modo	Codificación
Inmediato	`000000`
Directo	`001000`
Indirecto	`011000`
Registro	`100rrr`
Indirecto Registro	`110rrr`

1 Repeticiones

Mediante el uso de los saltos podemos mover el PC hacia atrás y lograr que una secuencia de instrucciones se ejecuten mas de una vez. Es importante que tengamos alguna forma de parar está repetición, ya que de lo contrario el programa entraría en un ciclo infinito, repitiendo y repitiendo dicha secuencia de instrucciones y no terminando nunca. En general esto lo hacemos con saltos condicionales, revisando alguna condición que puede cambiar cada vez que se ejecuta la secuencia de instrucciones. El esquema tipico de una repetición en Q es (ojo! pueden hacerse repeticiones de otras formas, este es un esquema a modo de ejemplo):

```
CICLO:CMP X, Y // X e Y pueden ser registros,
               // valores en memoria, etc
      JE FIN_CICLO // JE o cualquier
                   // salto condicional
      INSTRUCCIONES // Alguna de ellas
                    // debería modificar X o Y
      JMP CICLO
FIN_CICLO: // Continua el programa
```

Ejercicios

1.a Escribir **y documentar** una rutina que calcule la multiplicación R5*R6 sin usar la instrucción MUL. Consejo: $A * B$ es sumar A unas $B - veces$, por ejemplo $8 * 3$ es $8 + 8 + 8$.

1.b Escribir **y documentar** una rutina que calcule el la división entera R0%R1 sin usar la instrucción DIV.

1.c Escribir **y documentar** una rutina que cuente la cantidad de dígitos 1 en la cadena que está en R6.

1.d Escribir **y documentar** una rutina que calcula el factorial del valor almacenado en R5. dicho valor está representado en $BSS()$. El factorial de un número n es el producto de los números enteros positivos desde el 1 hasta n. Por ejemplo, el factorial de 5 es:

$$5! = 1 * 2 * 3 * 4 * 5$$

2 Modos indirectos

Los modos indirectos nos permiten acceder a memoria sin tener que utilizar una constante. Esto nos permite hacer programas que trabajen sobre varias posiciones de memoria de una manera mas facil. Por ejemplo si queremos trabajar con las posiciones 0000, 0001, 0002 y 00003 podemos poner el valor 0000 en un registro (por ejemplo R0), utilizar [R0] para acceder al valor y luego incrementar R0 en 1 de modo que ,en la siguiente iteración del ciclo, el registro *apunte* a la siguiente posición con la que se debe trabajar.

Ejercicios

2.a Considerar la instrucción `ADD [[0000]], R1`

(a) Ensamblar la instrucción.

(b) Completar el siguiente cuadro de uso de buses.

Etapa	Bus de control	Bus de dir.	Bus de datos

2.b Considerar la instrucción `MOV [[0000]], R1`

(a) Ensamblar la instrucción.

(b) Completar el siguiente cuadro de uso de buses.

Etapa	Bus de control	Bus de dir.	Bus de datos

2.c Ensamblar el siguiente programa, ubicándolo a partir de la celda `0FF0`

```
        MOV R1, [[0000]]
volver: CMP [R2], 0x0000
```

2.d Completar la siguiente tabla de accesos para el programa anterior.

Instrucción	**B.Inst.**	**B.Op.**	**Alm.Op.**

2.e Ensamblar el siguiente programa, ubicándolo a partir de la celda 0FF0

```
MOV R1, [[0x0000]]
MOV [R2], [R1]
MOV [[0x0000]], [[0x0001]]
ADD [R2], [R1]
ADD [[0x0000]], [[0x0001]]
```

2.f Completar la siguiente tabla de accesos para el programa anterior.

Instrucción	**B.Inst.**	**B.Op.**	**Alm.Op.**

2.g Considerando el siguiente mapa de memoria y valor de los registros. ¿Qué registro o posición de memoria se termina modificando en cada instrucción? ¿Cuál es el valor que queda almacenado?

	...
0000	0001
0001	0004
0002	0000
0003	0002
0004	0001
	...

Registro	Valor almacenado
R0	0x0001
R1	0x0003

Por ejemplo, para la instrucción:

```
MOV [R1], [[0x0002]]
```

La posición que se va a modificar es la 0x0003. El valor que queda almacenado es 0x0001 (vamos a la posición 0x0002 y ahi hay un 0x0000 y en esa posición hay un 0x0001).

(a) `MOV [R0], [[0x0003]]`

(b) `MOV [[0x0003]], [R0]`

(c) `MOV [[0x0003]], [[0x0001]`

(d) `MOV [[0x0001]], [R1]`

3 Arreglos

Un arreglo es un conjunto homogéneo de valores. Esto quiere decir que todos los valores tienen la misma naturaleza (el mismo tamaño y tipo). Los arreglos ocupan un bloque de posiciones de memoria consecutiva y sus valores pueden ocupar una sola celda cada uno, pero podrían ser mas grandes (varias celdas). El tamaño del arreglo es la cantidad de elementos que tiene (notar que no siempre es la cantidad de celdas que ocupa). Por ejemplo, el siguiente es un arreglo con las edades de 4 niños en un primer grado (en la posición 2000 está la edad de Juan, en la posición 2001 está la edad de María, y así siguiendo):

	...
2000	0005
2001	0006
2002	0006
2003	0005
	...

Como hariamos si tuvieramos un arreglo con las edades de todos los alumnos de orga y quisieramos calcular el promedio. Vamos a asumir que se les preguntó la edad, pero no necesariamente todos contestaron, por lo tanto no sabemos cuantas edades hay cargadas. En ese caso noos tienen que dar o bien la cantidad de alumnos encuestados o decirnos que luego de la última edad se cargo un valor especial. Por ejemplo, se podría cargar 0x0000 ya que ningún alumno tiene 0 años.

Entonces supongamos que nos dicen que a partir de la celda 0x2000 esta el arreglo y que va a terminar con 0x0000. Para ir recorriendo el arreglo podemos cargar la dirección 0x2000 en un registro y luego ir usando el modo indirecto a registro para trabajar sobre las posiciones, usemos R0 para dicha función.

Para calcular el promedio vamos a tener que por un lado sumar todas las edades y por otro contar cuantas edades sumamos. Usemos R1 para sumar las edades y R2.

```
MOV R0, 0x2000
MOV R1, 0 // La suma acumulada es 0
MOV R2, 0 // Contamos 0 alumnos
Ciclo:CMP [R0], 0X000 // Llegamos al final?
JE FIN
ADD R1, [R0] // Sumamos la edad
ADD R2, 0X0001 // Contamos uno mas
ADD R0, 0X0001 // Pasamos al siguiente alumno
JMP Ciclo
fin: DIV R1, R2 // R1 guarda el promedio
```

Ejercicios

3.1 Dado un arreglo que comienza en la posición `0x3333` y que tiene 5 elementos en $BSS(16)$, se pide escribir una rutina que sume los valores del arreglo.

3.2 Dado un arreglo que comienza en la posición de memoria cuya dirección se encuentra en R0 (es decir R0 tiene la direccón del primer elemento del arreglo y [R0] el valor de dicho elemento), y que termina con el primer elemento cuyo valor es 0x0000; se pide escribir una rutina que cuente la cantidad de elementos que tiene el arreglo.

3.3 Escribir una rutina que cuente cuantos números de un arreglo son pares. El arreglo comienza en la celda 4486 y la longitud del arreglo está en la celda 4485.

3.4 Escribir un programa que haga un control de calidad sobre la rutina anterior.

3.5 Escribir una rutina que cuente cuantos números de un arreglo tienen el quinto bit en 1. El arreglo comienza en la celda 4486 y la longitud del arreglo está en la celda 4485.

3.6 Escribir un programa que haga un control de calidad sobre la rutina anterior

3.7 (a) Implemente la siguiente rutina a partir de su documentación

```
;------------------------ maximoValorEnArregloABAB
; REQUIERE Un arreglo que comienza en la celda ABAB
;          Los valores ocupan 1 celda cada uno.
;          El tamaño del arreglo está en R7.
; MODIFICA ??
; RETORNA en R6 el valor máximo del arreglo
;-------------------------------------------------
```

(b) Modifique la rutina anterior para que reciba como parámetro la dirección inicial del arreglo en R0.

3.8 Implemente la siguiente rutina a partir de su documentación

```
;-----------------------------posDeMaximoEnArreglo
; REQUIERE En R5 la dirección inicial de un arreglo
;          En R7 el tamaño del arreglo.
;          Los valores ocupan 1 celda cada uno.
; MODIFICA ??
; RETORNA en R6 el la dirección del máximo
;-------------------------------------------------
```

3.9 Implemente la siguiente rutina a partir de su documentación

```
;-------------------------proyectarPosicionesPares
; REQUIERE En R5 la dirección inicial de un arreglo
;      Que el arreglo finalize con el valor FFFF
;      Los valores ocupan 1 celda cada uno.
;      En R4 la dirección inicial del nuevo arreglo
; MODIFICA ??
; RETORNA Un nuevo arreglo formado por los valores
;     de las posiciones pares del arreglo original.
;-----------------------------------------------
```

3.10 (a) Al finalizar la fabricación de cada par de zapatos, la máquina que lo produce almacena en una celda de memoria el valor 1 para indicar que el par se fabricó correctamente libre de errores o 0 para el caso contrario.

El siguiente mapa de memoria **es un ejemplo** de un día determinado de producción: Escribir un programa que utilice la subrutina `sumaSiEsUno` para contar la cantidad de pares de zapatos que se fabricaron correctamente, analizando los resultados que se volcaron en las celdas `2000` a `2003`. El total debe quedar en `R2`.

(b) Generalizar el programa anterior para un arreglo de cualquier longitud y cualquier posición, escribiendo una subrutina que reciba estos datos como parámetros. **¡Documéntela!**.

3.11 (a) En un centro de cómputos se realiza diariamente un chequeo de virus de las computadoras de la sala. Para llevar un registro de la actividad diaria se almacena en una celda de memoria el valor 0 para indicar que la PC se encuentra limpia y en caso contrario el número de virus encontrado. Escriba la siguiente rutina

```
;------------------------------contarCompusLimpias
; REQUIERE En R5 la dirección inicial de un arreglo
;      Que el arreglo finalize con el valor FFFF
;      Los valores ocupan 1 celda cada uno.
; MODIFICA ??
; RETORNA en R7 la cantidad de PC sin virus
;-------------------------------------------------
```

(b) Ejecute el siguiente programa

```
MOV R5, 0x1000
CALL contarcompusLimpias
```

Sabiendo que la rutina contarcompusLimpias esta ensamblada en 0xBB00 y que el siguiente es el estado de la memoria.

	...
1000	0000
1001	000A
1002	0003
1003	0000
1004	FFFF
	...

y completando la siguiente tabla de accesos:

Instrucción	B.Inst.	B.Op.	Alm.Op.

(c) Escriba la siguiente rutina, tomando como ejemplo la rutina `contarCompusLimpias`

```
;-----------------------------contarCompusVirusB0B0
; REQUIERE En R5 la dirección inicial de un arreglo
;      Que el arreglo finalize con el valor FFFF
;      Los valores ocupan 1 celda cada uno.
; MODIFICA ??
; RETORNA en R7 la cantidad de PC con el virus B0B0
;-------------------------------------------------
```

3.12 Escriba el programa `juventud` y la rutina `edadEnAnios` a partir de la información que se provee.

```
;---------------------------edadEnAnios
;REQUIERE 1) La fecha de nacimiento (dia,mes,año)
;            en los registros R5, R6 y R7 resp.
;         2) La fecha actual en los registros
;            R0, R1 y R2
;MODIFICA ?
;RETORNA en R7 la edad de la persona (en años)

;---------------------------juventud
;REQUIERE 1) La fecha de nacimiento de 3 personas,
;         almacenada cada una en 3 celdas
;         consecutivas a partir de la celda 7890
;         2) La fecha actual en los registros
;         R0, R1 y R2
;MODIFICA ?
;RETORNA  Un 1 en R7 si una o más de las 3
;         personas es mayor a 21
;         años y menor a 28, ó 0 en caso contrario
```

3.13 (a) Implemente la siguiente rutina a partir de su documentación

```
;-------------------- comparador
; REQUIERE Dos valores x e y en bss(16)
;          almacenados en los
;          registro R6 y R7
; MODIFICA ??
; RETORNA Si x<y suma 1 al registro R0
;         Si x=y suma 1 al registro R1
;         Si x>y suma 1 al registro R2
;--------------------
```

(b) Escriba un programa que cuente cuántos valores son menores y cuántos mayores a su predecesor en las celdas 7788 a 778B. Es decir, se deben hacer las comparaciones:

- la celda 7788 (x) con la celda 7789 (y)
- la celda 7789 (x) con la celda 778A (y)
- la celda 778A (x) con la celda 778B (y)

Los resultados deben volcarse en las celdas 6000 (menores) y 6001 (mayores).

(c) Generalice el programa anterior para los valores de un arreglo que comienza en 0345 y finaliza con el primer valor FFFF.

3.14 (a) Implemente la siguiente rutina

```
;-------------------- fibonacci
; REQUIERE Un valor n en bss(16) almacenado
```

```
;          en el registro R7
; MODIFICA ??
; RETORNA El n-esimo valor de la serie de
;         Fibonacci en el registro R6
;--------------------
```

Cada número en la serie de Fibonacci se define como la suma de los dos números anteriores, es decir:

$$0, 1, 1, 2, 3, 5, 8, 13...$$

Entonces, si se ejecuta el siguiente programa

```
MOV R7, 0x0005
CALL fibonacci
```

se obtiene el valor 3 en el registro R6.

(b) Escriba un programa que cargue las celdas 7788 a 778B con los primeros 4 números de la serie de Fibonacci

3.15 (a) En una fábrica de ventanas las características de los productos de cada venta se codifican mediante cadenas de 16 bits. Cada bit representa una cualidad y se coloca un 1 si cumple con dicha característica, por ejemplo:

- Con el **bit 3** se indica si está pintada
- Con el **bit 9** se indica si lleva el vidrio de seguridad

(**Nota**: Tener en cuenta que los demás bits también representan distintas características, pero para el ejercicio solo nos interesan esos dos).

En caso que la ventana esté pintada o lleve vidrio de seguridad es necesario usar un embalaje distinto, y para esto se pide escribir un programa que determine colocando un 1 en R7 si el pedido que está en la celda `0019` requiere un embalaje *premium*, es decir, si se trata de una ventana pintada o con vidrio de seguridad.

(b) Se tiene un arreglo de pedidos a partir de la celda `0001`, cuya longitud está en en la celda `0000`. Escriba un programa que recorra el arreglo para contar en R6 los pedidos que **no necesitan embalaje *premium* segun el ejercicio anterior**.

3.16 (a) En un arreglo se tienen codificados los pedidos de una rotisería en cadenas de 16 bits, donde el byte mas significativo representa el tipo de producto y el byte menos significativo la cantidad de unidades. Los códigos de producto son:

- Si bit 15 (mas significativo) es 1: empanadas de carne
- Si bit 14 es 1: empanadas de pollo
- Si bit 13 es 1: empanadas de jamón y queso
- Si bit 12 es 1: pizza napolitana

Por ejemplo:

Cadena	**Interpretación**
1000000 00000110	6 empanadas de carne
0010000 00001010	10 empanadas de jamón y queso

Se necesita calcular el trabajo de los cocineros, totalizando la cantidad de empanadas de cada tipo. Escriba las siguientes subrutinas:

contarEmpCarne Cuenta en R4 la cantidad pedida de empanadas de carne

contarEmpPollo Cuenta en R5 la cantidad pedida de empanadas de pollo

contarEmpJyQ Cuenta en R6 la cantidad pedida de empanadas de jamón y queso

contarPizzaNapo Cuenta en R5 la cantidad pedida de pizzas napolitanas

Considere que el arreglo de pedidos comienza en la celda 5310, y termina con el primer valor 0.

(b) Considerando la rotisería descripta en el ejercicio anterior, escriba un programa que calcule la ganancia a partir del arreglo de pedidos y un arreglo de precios unitarios que ocupa el rango de celdas 5200..5203 (pues son 4 productos).

4 Controlar los programas

Para probar las rutinas/programas que escribiste, escribiremos programas de *test* o prueba como se explicó en la práctica de máscaras

4.a Escribí un programa que haga un control de calidad sobre la rutina `maximoValorEnArregloABAB`

4.b Escribí un programa que haga un control de calidad sobre la rutina `posDeMaximoEnArreglo`

4.c Escribí un programa que haga un control de calidad sobre la rutina `factorial`