Guía de ejercicios # 10: Iteraciones, arreglos y recorridos

Organización de Computadoras 2016

UNQ

Arquitectura Q6

Características

- Tiene 8 registros de uso general de 16 bits: R0..R7.
- La memoria utiliza direcciones de 16 bits.
- Tiene un 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 códigos de operaciones		
Operación	Cod Op	Efecto
MUL	0000	Dest ← Dest * Origen [1]
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 [2]

Instrucciones de un operando origen

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

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

[1] El resultado de la operación `MUL` ocupa 32 bits, almacenándose los 16 bits menos significativos en el operando destino y los 16 bits mas significativos en el registro `R7`

[2] El caracter % denota el cociente de la división entera

Instrucciones de un operando destino

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

Tabla de códigos de operaciones		
Operación	Cod Op	Efecto
NOT	1001	Dest ← ¬Dest

Instrucciones sin operandos

Formato de Instrucción	
CodOp (4b)	Relleno (000000000000)

Tabla de códigos de operaciones		
Operación	CodOp	Efecto
RET	1100	SP ← SP + 1; PC ← [SP]

Saltos condicionales

Cod_Op (8)	Desplazamiento(8)

donde 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`[3]
Registro	`110rrr`[4]

1 Repeticiones

1. Escribir **y documentar** una rutina que cuente la cantidad de dígitos 1 en la cadena que está en R6
2. Escribir **y documentar** una rutina que calcule la multiplicación R5*R6 sin usar la instrucción MUL.
3. Escribir **y documentar** una rutina que calcule el la división entera R0%R1 sin usar la instrucción DIV.
4. Escribir **y documentar** una rutina `sumaSiEsUno` que incrementa en 1 el registro R6 si el valor de R5 es 1, y no hace nada en caso contrario.
5. 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

1. Ensamblar la instrucción `MOV [[0000]], R1`
2. ¿Cuantos accesos se llevan a cabo durante la ejecución de la instrucción anterior? Distinguir las etapas de búsqueda de instrucción, búsqueda de operandos y almacenamiento de operandos.
3. Ensamblar el siguiente programa, ubicándolo a partir de la celda 0FF0

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

4. Completar la siguiente tabla de accesos para el programa anterior.

Instrucción	B.Inst.	B.Op.	Alm.Op.

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

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

[3] `rrr` es una codificación (en 3 bits) del número de registro
[4] `rrr` es una codificación (en 3 bits) del número de registro

6. Completar la siguiente tabla de accesos para el programa anterior.

Instrucción	B.Inst.	B.Op.	Alm.Op.

3 Arreglos

1. 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:

	...
2000	0000
2001	0001
2002	0001
2003	0000
	...

   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`.

2. Generalizar el programa anterior para un arreglo de cualquier longitud y cualquier posición, escribiendo una subrutina que reciba estos datos como parámetros. **¡Documéntela!**.
3. 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
;-------------------------------------------------
```

4. 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.

5. Escriba la siguiente rutina, tomando como ejemplo la rutina `contarCompusLimpias` del ejercicio 13

```
;----------------------------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
;-------------------------------------------------
```

6. 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
```

7. 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
;--------------------
```

8. 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).

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

10. 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
;-------------------------------------------------
```

11. Modifique la rutina anterior para que reciba como parámetro la dirección inicial del arreglo.

12. Implemente la siguiente rutina a partir de su documentación

```
;-----------------------------posDeMaximoVEnArreglo
; 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
;-------------------------------------------------
```

13. 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.
;-------------------------------------------------
```

14. 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.

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

16. Generalizar el programa anterior en una rutina

```
;---------------------------------generarFibonacci
; REQUIERE En R5 la dirección inicial de un arreglo
;      En R4 el tamaño del arreglo
; MODIFICA Las posiciones del arreglo, cargándolas
;      con los números de fibonacci.
;-------------------------------------------------
```

17. Documente la siguiente rutina

```
              MOV R1, [1020] ; long. del arreglo A
              MOV R0, 0x1021 ; se ubica al inicio
                             ; del arreglo A
              MOV R7, 0x3021 ; se ubica al inicio
                             ; del arreglo resultado
      arriba: CMP R1, 0x0000
              JE finA
              MOV [R7], 0x000; inicializa el contador
                             ; correspondiente
              MOV R2, 0x2021 ; se ubica al inicio
                             ; del arreglo B
              MOV R3, [2020] ; longitud del
                             ; arreglo B
       busca: CMP R3, 0x0000
              JE finB
              CMP [R0], [R2]
              JNE nosoniguales
              ADD [R7], 0x0001
nosoniguales: ADD R2, 0x0001; sig. elemento
                           ; de B
              SUB R3, 0x0001
              JMP busca
        finB: ADD R0, 0x0001; sig. elemento
                           ; de A
              SUB R1, 0x0001
              ADD R7, 0x0001; sig. elemento
                           ; del resultado
              JMP arriba
        finA: RET
```

18. Dado el siguiente par de arreglos, ¿Cómo queda la memoria luego de ejecutar la rutina del ejercicio 16?

	...
1020	0002
1021	1111
1022	0AA3
⋮	⋮
2020	0004
2021	1111
2022	0AA3
2023	078E
2024	0AA3
	...

19. Modificar el programa del ejercicio 16 para que el arreglo resultado quede invertido.

20. Escribir un programa que cuente cuantos números de un arreglo son pares y cuántos tienen el bit 5 en 1. El arreglo comienza en la celda 4486 y la longitud del arreglo está en la celda 4485.

21. 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.

22. 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**.

23. 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.

24. 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).