Guía de ejercicios # 4
Rutinas: modularización y reuso

Organización de Computadoras 2018

UNQ

Los objetivos de esta práctica son:

- Poder dividir en sub tareas un problema
- Volcar un sub problema en una rutina utilizando para su programación Q3
- Realizar correctamente la documentación de las rutinas programadas
- Invocar rutinas sabiendo o no como estan programadas (en el caso de no saber, leyendo sólo su documentación)
- Entender como funciona una pila y las instrucciones CALL y RET en la arquitectura Q reflejando correctamente los cambios en los registros especiales y la pila Ensamblar y desensamblar programas en Q3

Los ejercicios marcados con ★ son un conjunto minimal para comprender los temas tratados en esta práctica. Para resolver esta práctica se aconseja consultar el apunte de la materia *Modularización y Reuso: Rutinas en Q3*, disponible en `http://orga.blog.unq.edu.ar/descargas/`

Arquitectura Q3

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`.

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
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

Instrucciones sin operandos

Formato de Instrucción

CodOp (4b)	Relleno (000000000000)

Tabla de instrucciones

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

Modos de direccionamiento

Modo	Codificación
Inmediato	`000000`
Directo	`001000`
Registro	`100rrr`

1 Implementación e invocación de rutinas

- Ejemplo de cómo implementar y documentar una rutina según su especificación:

  Escriba una rutina `divPorNCuadrado`, que divida el contenido de de `R2` por un número n recibido por párametro y eleve el resultado al cuadrado. **Documente la rutina** especificando requiere, retorna y modifica

```
# ---------- Rutina divPorNCuadrado ------------
# Requiere: El divisor "n" en el registro R1
# El dividendo en el registro R2
# Modifica: R2
# Retorna:  El resultado de la división
entre el dividendo y el divisor
            elevado al cuadrado en R3

divPorNCuadrado: DIV R2, R1
                 MUL R2, R2
                 MOV R3, R2
                 RET
```

- Ejemplo de cómo invocar una rutina preexistente dentro de un programa:

  Utilice la rutina `divPorNCuadrado` para escribir un **programa** que calcule 12 dividido 4 elevado al cuadrado y lo guarde en el registro `R1`

```
MOV R2, 0x000C
MOV R1, 0x0004
CALL divPorNCuadrado
MOV R1, R3
```

1. Escriba una rutina `mulPorDos`, que multiplique por 2 el contenido de `R1` y guarde el resultado en `R1` ★
   Documente la rutina especificando requiere, retorna y modifica. ★

2. Utilice la rutina `mulPorDos` para escribir un **programa** que calcule 2 elevado a la 5. ★

3. Escriba una rutina `mulPorCuatro`, que multiplique por 4 el contenido de `R1` y guarde el resultado en `R1` . **Documente la rutina** especificando requiere, retorna y modifica.

4. Utilice la rutina `mulPorCuatro` para escribir un **programa** que calcule 4 elevado a la 4.

5. Escriba una rutina `aLaQuinta`, que eleve a la 5ta potencia el contenido del registro `R0` . **Documente la rutina** especificando requiere, retorna y modifica.

6. Utilizando la rutina `aLaQuinta`, escriba un programa que calcule n^5 para los números que están almacenados en la memoria desde la celda `0x1000` hasta la celda `0x1007` **inclusive**.

7. Escriba una rutina `esPar`, que determine si el contenido de `R0` es par o impar de la siguiente manera: Si es par, debe guardar un 0 en `R1` , en caso contrario, debe guardar un 1. **Documente la rutina** especificando requiere, retorna y modifica ★

8. Utilizando la rutina `esPar`, escriba un programa que grabe un 0 o un 1 en `0x0000`, ..., `0x0005` según si los números de `0xF000`, ..., `0xF005` son pares o no ★

9. Escribir una rutina `swapR0R1` que intercambie los valores de los registros R0 y R1. Documente su rutina★

10. Escribir un programa que utilice la rutina `swapR0R1` para intercambiar los valores de las celdas consecutivas entre la 0x3000 y la 0x3009. Esto es: intercambiar el valor de **X** con el valor de **X**+1, siendo **X** una celda cuya dirección es par, en el rango [0x3000, 0x3009] ★

11. Escribir una rutina `avg` que calcule el promedio entre R1 y R2, guardando el calculo en R3. **Documente la rutina** especificando requiere, retorna y modifica.

12. Sabiendo que en R1, R2, R4 y R5 se encuentran almacenadas las edades de los profes de ORGA, calcular el promedio total utilizando la rutina `avg`.

13. Se cuenta con la subrutina `maxInt` que calcula el maximo entre los valores $BSS(16)$ de R6 y R7, dejando el resultado en R6.
    Escribir un programa que calcule el maximo de los registros R1 al R7 **inclusive** ★

14. Se cuenta con las siguientes documentaciones de la rutinas `sumados` y `aplicarDescuento`:

```
; SUMADOS
; REQUIERE: Dos valores a sumar almacenados
            en R1 y R2
; MODIFICA: R1
; RETORNA: En R1 la suma de los valores
           almacenados en R1 y R2.


; APLICARDESCUENTO
; REQUIERE: El precio unitario en la celda 0xA000
;           El porcentaje a aplicar
;                 en la celda 0xA001
; MODIFICA: R0
; RETORNA: El precio con el descuento
;          aplicado en R0.
```

    Utilizando las rutinas `aplicarDescuento` y `sumaDos`, escriba un programa que calcule el precio final a pagar por una persona que compra dos productos, cuyos precios unitarios están almacenados en R6 y R7 y se le debe aplicar un descuento del 10% ★

15. Dada la siguiente secuencia de números 2, 4, 6, 8, 10 escribir un programa que calcule la sumatoria utilizando la rutina `sumaDos`.

16. Responder las siguientes preguntas teniendo en cuenta lo ejercitado en esta sección ★

    Dado un programa que quiere invocar a una rutina:

    (a) ¿Dónde debe estar la instrucción CALL: en el programa o en la rutina?

    (b) ¿Dónde debe estar la instrucción RET: en el programa o en la rutina?

    (c) ¿Hace falta saber como esta programada una rutina para poder utilizarla correctamente? ¿Por qué?

2 Simular la ejecución de rutinas

17. Considere la siguiente rutina:

```
rutina1: MOV R1, R0
         RET
```

    y el siguiente programa:

```
programa: CALL rutina1
          CALL rutina1
```

    Sabiendo que:

- `rutina1` está ensamblada a partir de la celda `0x00E0`
- el programa está ensamblado a partir de la celda `0x1000`
- PC=1000
- La pila está vacía.

Simule los cambios que ocurren en el PC, en el SP y en el contenido de la pila completando una tabla de ejecución como la que sigue (ver apunte)

PC	Instruccion	PC-BI	PC-Ex	SP	Pila
1000	CALL rutina1				

18. Considere las siguientes rutinas:

```
rutina1: MOV R1, R0
         RET

rutina2: RET
```

y el siguiente programa:

```
programa: CALL rutina1
          CALL rutina2
```

Sabiendo que:

- `rutina1` está ensamblada a partir de la celda `0x00E0`
- `rutina2` está ensamblada a partir de la celda `0x00A1`
- el programa está ensamblado a partir de la celda `0x1000`
- PC=`0x1000`
- La pila está vacía.

Simule los cambios que ocurren en el PC, en el SP y en el contenido de la pila completando la tabla de ejecución (ver apunte). ★

19. Considere las siguientes rutinas:

```
rutina1: MOV R0, R1
         CALL rutina2
         RET

rutina2: MOV R2, R3
         RET
```

Y el siguiente programa:

```
principal: CALL rutina2
           CALL rutina1
```

Sabiendo que:

- `rutina1` está ensamblada a partir de la celda `0x00F0`
- `rutina2` está ensamblada a partir de la celda `0x00A1`
- el programa está ensamblado a partir de la celda `0x0E00`
- PC=`0x0E00`
- La pila está vacía.

¿Cómo varía la pila durante la ejecución del programa `principal`?

20. Considere las siguientes rutinas:

```
rutina1: MOV R1, R0
         CALL rutina2
         RET

rutina2: CALL rutina3
         RET

rutina3: MOV R2, R1
         RET
```

y el siguiente programa:

```
programa: CALL rutina1
          CALL rutina2
          CALL rutina3
```

Sabiendo que:

- `rutina1` está ensamblada a partir de la celda `0x00E0`
- `rutina2` está ensamblada a partir de la celda `0x00A1`
- `rutina3` está ensamblada a partir de la celda `0x0101`
- el programa está ensamblado a partir de la celda `0x1000`
- PC=0x1000
- La pila está vacía.

Simule los cambios que ocurren en el PC, en el SP y en el contenido de la pila completando la tabla de ejecución (ver apunte).

21. Dado el siguiente programa:

```
programa: SUB R0, 0x0001
          CALL rutina
          DIV R5, 0x0002
```

La siguiente rutina:

```
rutina: MUL R0, 0X0003
        MOV R5, R0
        RET
```

Y teniendo los siguientes datos:

- El programa esta ensamblado a partir de la celda 0x0020
- La rutina esta ensamblada a partir de la celda 0x0E00

- La pila está vacía.

Encuentre los errores en la tabla de ejecución ★

PC	Instruccion	PC-BI	PC-Ex	SP	Pila
0x0020	SUB R0,0x0001	0x0021	0x0022	0xFFEF	[]
0x0022	CALL rutina	0x0024	0x0E00	0xFFEE	[0x0024]
0x0E00	MUL R0,0X0003	0x0E02	0x0E02	0xFFEE	[0x0024]
0x0E02	RET	0x0E03	0x0024	0xFFEE	[]
0x0E02	DIV R5,0x0002	0x0E04	0x0E04	0xFFEF	[]

3 Ensamblado de programas

Para los ejercicios de ensamblado de programas tener en cuenta:

- Los programas y las rutinas estan ensamblados a partir de direcciones de memoria diferentes (la rutina no esta ensamblada dentro del programa aunque desde allí se la invoque).
- Cada etiqueta se traduce en un valor por lo cual el modo de direccionamento es inmediato (el valor es la dirección de memoria a partir de la cual esta ensamblada la rutina que lleva el nombre de la etiqueta).
- No olvidar los rellenos en el ensamblado de las instrucciones, tanto de un operando de origen y sin operandos.

22. Ensamblar el siguiente programa a partir de la celda `0xFF0E` ★

```
SUB  R0, 0x0001
CALL restarTriple
MOV  R3, [0x0A0A]
ADD  R3, R0

restarTriple: MOV R1, R0
              MUL R1, 0x0003
              SUB R0, R1
              RET
```

Sabiendo que `restarTriple` se encuentra ensamblado a partir de la celda `0x1000` ensamblar dicha rutina también.

23. Corroborar que el programa y la rutina fueron ensamblados correctamente desensamblando el código máquina que obtuvo como resultado. ★

24. Ensamblar el siguiente programa a partir de la celda `0xA010`

```
MUL  R3, 0x0005
CALL restarMitad
SUB  R3, 0X0079

restarMitad: MOV R2, R3
             DIV R2, 0x0002
             SUB R3, R2
             RET
```

Sabiendo que `restarMitad` se encuentra ensamblado a partir de la celda `0xF120` ensamblar dicha rutina también.

25. Corroborar que el programa y la rutina fueron ensamblados correctamente desensamblando el código máquina que obtuvo como resultado.

4 Ejercicios para utilizar QSim

26. Utilizando QSim, ejecute paso a paso el siguiente programa, observando el comportamiento de la pila y el registro SP. Indique con qué valor queda el registro R2 al finalizar la ejecución de la cuarta instrucción (`MOV R2, R0`):

```
MOV R0, 0x000A
MOV R1, 0x000B
CALL sumamult
MOV R2, R0

sumamult:
   ADD R0, 0x000C
   ADD R1, 0x000C
   CALL mult
   RET

mult:
   MUL R0, R1
   RET
```

27. En el ejercicio anterior, ¿cómo se traducen las etiquetas *sumamult* y *mult*? Es decir, que valor tiene cada una teniendo en cuenta que deben ser el número de celda en la memoria donde comienzan a estar ensambladas cada una de las rutinas respectivamente.

28. Pruebe en QSim, qué efecto tiene utilizar CALL con modos de direccionamiento registro o directo en lugar de una etiqueta ★

Por ejemplo:

```
MOV [0x000A], 0x001A
CALL [0x000A]


MOV R0, 0x000A
CALL R0
```

Intentar con todos los modos de direccionamiento soportados por la instrucción CALL (Tener en cuenta las respuestas de la sección anterior **Funcionamiento de las instrucciones CALL y RET**)

29. Utilizando QSim, pruebe el siguiente programa:

```
MOV R0, 0x0001
MOV R1, 0x0002
CALL factInfinito
```

```
factInfinito:
   MUL R0, R1
   ADD R1, 0x0001
   CALL factInfinito
```

Luego responda las siguientes preguntas:

- ¿Qué se está calculando en el registro R0?
- ¿Qué defecto tiene el programa?
- ¿Cómo crece la pila? ¿Qué valores se apilan?
- ¿Dada la arquitectura de 16 bits de QSim, hasta qué punto tiene sentido que el programa continue calculando?

References

[1] Rutinas y su implementación , http://orga.blog.unq.edu.ar/wp-content/uploads/sites/5/2015/08/orga_apunte_rutinas-1.pdf