Práctica 3: Arquitectura del CPU

Programación en Assembler de ORGA1

Andrea V. Manna

Organización del Computador I
DC - UBA

1er Cuatrimestre 2018

Acordando pautas...

- Se utilizará el lenguaje Assembler de la máquina ’Orga 1’
- Es un lenguaje ensamblador particular de esta arquitectura, que es una simplificación de una arquitectura Intel x86
- Posee instrucciones con 2 operandos, 1 operando, 0 operandos y saltos
- Posee diferentes modos de direccionamiento

Enunciado Ejercicio Nro 1

Un clásico para empezar...Como en memoria tenemos "tiras"de bits, se me ocurre tener un vector de enteros de 16 bits almacenado a partir de la dirección identificada por la etiqueta **VECTOR**. Hagamos entonces una rutina que calcule la cantidad de ceros del vector. El tamaño del vector se encuentra en la posición de memoria identificada por la etiqueta **SIZE**. Vamos a retornar el resultado en la posición de memoria identificada por la etiqueta **CEROS**.

Pseudocódigo

Inicializar Registros
para cada elemento del vector
    si vector(i) == 0 **entonces**
        incrementar el contador de ceros
    fin si
fin para
Colocar el resultado en la etiqueta **CEROS**

Assembler!

```
  mov R0, VECTOR ;Cargo en R0 la dirección de comienzo del vector
  mov R1, 0x0000 ;En R1 cuento la cantidad de ceros.Lo inicializo en 0
  mov R2, [SIZE] ;En R2 cuento la cantidad de elementos por recorrer.
                 ;Lo inicializo con el tamaño del vector
ciclo: cmp [R0], 0x000 ;Comparo al 0 con el elemento del vector actual
                       ;(el apuntado por R0)
  jne seguir     ;Si no eso cero salto a la etiqueta seguir para
                 ;saltearme la instrucción que incrementa a R0
  add R1, 0x0001 ;Si es cero incremento R1 en 1
seguir: add R0, 0x0001 ;Actualizo R0 para que apunte al
                       ;siguiente elemento del vector
  sub R2, 0x0001 ;Decremento R2 indicando que falta un
                 ;elemento menos por recorrer
  jne ciclo      ;Si quedan elementos por recorrer vuelvo a ciclo
  mov [CEROS], R1;Termine de recorrer el vector.
                 ;Ahora almaceno el resultado en CEROS
fin:
```

Enunciado Ejercicio Nro 2

Estaría muy bueno manejar algunas operaciones en Assembler. Y la división tiene sus vericuetos....Por lo tanto vamos a escribir un programa que calcule la división entera entre dos enteros sin signo de 16 bits.

- R1 contiene la dirección de memoria donde se aloja el dividendo.
- R2 contiene la dirección de memoria donde se aloja el divisor.
- R3 debe ser el registro en el que se devuelva el cociente (el resultado de la división).

Como somos muy buenos...En caso de que el divisor sea 0, basta con que se retorne 0.

Pseudocódigo

```
resultado = 0
si divisor == 0 entonces
    listo
sino
    mientras dividendo >= divisor hacer
        dividendo = dividendo - divisor
        resultado = resultado + 1
    fin mientras
fin si
```

Assembler!

```
; R1 --> puntero al dividendo
; R2 --> puntero al divisor
; R3 --> cociente
; R4 --> dividendo
; R5 --> divisor
inicio:
        MOV R3, 0x0000 ; R3 = 0
        MOV R4, [R1]   ; R4 = dividendo
        MOV R5, [R2]   ; R5 = divisor
        CMP R5, 0x0000 ; divisor == 0?
        JE fin
ciclo:  CMP R4, R5 ; dividendo < divisor?
        JCS fin    ; uso JCS en vez de JL porque son enteros sin signo
        SUB R4, R5     ; R4 = R4-R5
        ADD R3, 0x0001 ; R3 = R3+1
        JMP ciclo
fin:
```

Tarea: Retornar en R0 el resto de la division!

Enunciado Ejercicio Nro 3

Los docentes del turno mañana precisamos ingresos extras... compramos una pareja de conejos y queremos saber cuántos se podrían reproducir en un año a partir de la pareja inicial, teniendo en cuenta que de forma natural tienen una pareja en un mes, y que a partir del segundo mes se empiezan a reproducir.

Sucesión de Fibonacci

$$f_n = f_{n-1} + f_{n-2} \text{ con } f_0=1 \text{ y } f_1=1$$

Pseudocódigo

```
meses = 10; //(12-2) me da el término 12 de la suc.
anteUltMes = 1;
ultMes = 1;
mientras meses > 0
    cantParejas = ultMes;
    cantParejas = cantParejas + anteUltMes;
    anteUltMes = ultMes;
    ultMes = cantParejas;
    meses = meses - 1;
fin mientras
```

Assembler!

```
; R0 --> meses
; R1 --> anteUltMes
; R2 --> ultMes
; R3 --> cantParejas
inicio: MOV R0, 0x000A    ; meses = 10
        MOV R1, 0x0001    ; anteUltMes = 1
        MOV R2, 0x0001    ; ultMes = 1
ciclo:  MOV R3, R2        ; cantParejas = ultMes;
        ADD R3, R1        ; cantParejas += anteUltMes;
        MOV R1, R2        ; anteUltMes = ultMes;
        MOV R2, R3        ; ultMes = cantParejas;
        SUB R0, 0x0001    ; meses = meses - 1;
        JG ciclo          ;
fin:
```

Conclusiones

- Vimos como realizar pequeños programas en Assembler utilizando la Arquitectura ’Orga 1’
- Es importante utilizar pseudocódigo para resolver el problema y verificar que el algoritmo funcione
- Pueden utilizar un pseudocódigo informal, que pueda comprenderse y no sea ambiguo
- Una vez verificada la solución, realizar el código en el lenguaje Ensamblador de la materia