Práctica 6: lenguaje ensamblador

Organización y Arquitectura de Computadoras

Hernández Ferreiro Enrique Ehécatl
López Soto Ramses Antonio

31 de marzo de 2019

1 Introducción

El **lenguaje ensamblador** es la aquella herramienta que nos brinda la representación simbólica de la codificación binaria de la computadora, es decir, en **lenguaje de máquina**.

El **ensamblador** el el encargado de traducir el lenguaje ensamblador a instrcciones binarias, el cual sólo lee un *único* archivo fuente y produce un *archivo objeto* que contiene instrucciones de máquina e información que ayuda a combinar varios objetos en un programa.

El lenguaje ensamblador toma dos papeles:

- Es el lenguaje de salida de los compiladores.
- Es un lenguaje más con el cual es posible programar.

2 Desarrollo

La práctica consistió en desarrollar pequeños programas mediante el uso del software *MARS* que nos presenta la arquitectura MIPS.

Se implemtaron 3 ejercicios que consitían en:

- Copiar el contenido de un registro a otro.
- Obtener el máximo común divisor de dos números.
- Obtener el cociente y residuo de una división.

Aqui podemos ver como copiar el contenido de un registro a otro:

```
.data
.text
        add $t1, $t1, 4
        add $v0, $t1, 0
        mul $t1, $t1, 0
        add $t0, $t1, $t2
```

El máximo común divisor se muestra como sigue:

```
.data
        numA: .ascii "Inserte numero a \n"
        numB: .ascii "Inserte numero b \n"
        RMCD: .ascii "El MCD es: "

.text

        li $v0, 4
        la $a0, numA
        syscall
        li $v0, 5
        syscall                 #Solicitud del num A
        add $t0, $v0, 0         #move


        li $v0, 4
        la $a0, numB
        syscall
        li $v0, 5
        syscall                 #Solicitud del num B
        add $t1, $v0, 0         #move

CalMCD:
        add $t2, $t1, 0         #move
        div $t7, $t0, $t1
        mfhi $t3
        add $t0, $t1, 0         #move
        add $t1, $t3, 0         #move

MCD:
        bnez $t3, CalMCD        #Branch if no 0

        li $v0,4
        la $a0, RMCD
```

```
        syscall
        li $v0, 1               #Imprime el resultado
        add $a0, $t2, 0         #move
        syscall
```

Y, finalmente, el cociente y el residuo de una divisíon es:

```
.data
        numA: .ascii "Inserte numero a \n"
        numB: .ascii "Inserte numero b \n"

.text

        li $v0, 4
        la $a0, numA
        syscall
        li $v0, 5
        syscall                 #Solicitud del num A
        add $t0, $v0, 0         #move

        li $v0, 4
        la $a0, numB
        syscall
        li $v0, 5
        syscall                 #Solicitud del num B
        add $t1, $v0, 0         #move

        add $k0, $k0, 0
        add $k1, $k1, 0

x2:
        subu $s1, $t0, $t1      #Resta t0 con t1
        add $k0, $k0, 1         #Contador de iteraciones

ciclo:

        move $t0, $s1           #Resta hasta que el resultado sea <= 0
        bge $s1, 0, x2

res:

        add $v1, $s1, $t1       #Suma el ultimo resultado de la resta con t1
        mul $v0, $zero, 0       #limpiar v0
```

```
sub $k0, $k0, 1          #restar uno al contador
add $v0, $v0, $k0        #Guarda el cociente (# de iteraciones)
```

3 Conclusión

3.1 Preguntas

1. ¿Existe alguna diferencia en escribir programas en lenguaje ensamblador comparado con un lenguaje de alto nivel?

   Los lenguajes de alto nivel se diseñaron con la finalidad de eliminar las cosas particulares de una computadora en específico, en cambio en el lenguaje ensamblador se diseño para una sola computadora en especifico o mas bien para un conjunto de microprocesadores.

2. ¿En qué caso es preferible escribir programas en lenguaje ensamblador y en qué casos en preferible hacerlo con un lenguaje de alto nivel?

   - Para Lenguaje ensamblador:
     Si se desea escribir un programa en el que se requiere que no se utilice mucha memoria y el tiempo de ejecución mucho menor, es mejor utilizar el lenguaje ensamblador.
   - Para lenguaje de alto nivel:
     Si se desea programar de uan manera más cómoda y, además, depurar el código fácilmente, es mucho mejor utilizar los lenguajes de alto nivel.