

Instrucciones

Tipo R

```
add $0, $1, $2
```

Almacena en el registro `$0` la **suma** de `$1` y `$2`

Tipo I

```
lw $1, d($2) // d = 0 / 4 / 8 ...
```

Carga en el registro `$1` la palabra almacenada en la dirección de memoria que contiene el registro `$2` más el **desplazamiento**. La dirección calculada debe ser múltiplo de 4.

```
sw $1, d($2)
```

Almacena en memoria la palabra del registro `$1` en la dirección de memoria que contiene el registro `$2` más el **desplazamiento**. La dirección debe ser múltiplo de 4.

```
addi $1, $2, 3
```

Almacena en el registro `$1` la **suma** del registro `$2` y la constante

```
beq $1, $2, tag
```

Si el valor de `$1` y `$2` es igual se **modifica el valor del PC** para pasar a ejecutar el trozo de código apuntado por la etiqueta

```
bne $1, $2, tag
```

Si el valor de `$1` y `$2` NO es igual se **modifica el valor del PC** para pasar a ejecutar el trozo de código apuntado por la etiqueta

Tipo J

```
j tag
```

Modifica el valor del PC para ejecutar la instrucción siguiente a la etiqueta

Pseudoinstrucción `la`

```
.data
cadena: .asciiz "Hola mundo"

la $a0, cadena
```

Guarda en `$a0` la dirección a la etiqueta (`cadena`). Es una pseudoinstrucción, se divide en:

```
lui $a0, 0x1001
ori $a0, $a0, 0x0000
```

Syscall

```
syscall
```

Toma el tipo de llamada al sistema del registro `$v0` y los argumentos de los registros `$a0` y `$a1` . Por ejemplo

```
addi $v0, $0, 10 // Añadimos al registro $v0 el código de la instrucción
syscall
```

Código	Función
10	Finalizar la ejecución
1	Imprime como entero lo que se encuentre en <code>\$a0</code>
4	Imprime como string lo que se encuentre en la dirección de memoriao indicada por <code>\$a0</code>

Secciones

Las primeras líneas de la sección `.text` son obligatorias ya que indican el punto de inicio del programa:

```
.text
.globl main

main: // A partir de esta llamada se encontrarían nuestras instrucciones
```

La sección `.data` contiene las variables del programa y es opcional. Estas se declaran de la siguiente forma:

```
.data
[ETIQUETA] : [TIPO] [INICIALIZACION]
cadena: .asciiz "Hola mundo"
```

Tipo	Definición	Estructura de la definición
.ascii	Almacena en memoria el string como una lista de caracteres	variable: .ascii "string a almacenar"
.asciiz	Almacena en memoria el string como una lista de caracteres y lo termina con 0	variable: .asciiz "string a almacenar"
.word	Almacena la lista de palabras en posiciones secuenciales de memoria	variable: .word palabra1, palabra2, ...
.space	Reserva n bytes de espacio en la memoria	variable: .space n
.float	Almacena la lista de números en punto flotante de simple precisión en posiciones sucesivas de memoria	variable: .float f1, f2, ...
.double	Almacena la lista de números en punto flotante de doble precisión en posiciones sucesivas de memoria	variable: .double d1, d2, ...

Simulador

Registros

PC=00400000 EPC=00000000 Cause=00000000
Status=0000ff03 BadVaddr=00000000
HI=00000000 LO=00000000

Registros generales			
\$zero=00000000	\$t0=00000000	\$s0=00000000	\$t8=00000000
\$at=00000000	\$t1=00000000	\$s1=00000000	\$t9=00000000
\$v0=00000000	\$t2=00000000	\$s2=00000000	\$k0=00000000
\$v1=00000000	\$t3=00000000	\$s3=00000000	\$k1=00000000
\$a0=00000000	\$t4=00000000	\$s4=00000000	\$gp=00000000
\$a1=00000000	\$t5=00000000	\$s5=00000000	\$sp=70000000
\$a2=00000000	\$t6=00000000	\$s6=00000000	\$fp=00000000
\$a3=00000000	\$t7=00000000	\$s7=00000000	\$ra=00000000

Registros de punto flotante			
\$f0=00000000	\$f8=00000000	\$f16=00000000	\$f24=00000000
\$f1=00000000	\$f9=00000000	\$f17=00000000	\$f25=00000000
\$f2=00000000	\$f10=00000000	\$f18=00000000	\$f26=00000000
\$f3=00000000	\$f11=00000000	\$f19=00000000	\$f27=00000000
\$f4=00000000	\$f12=00000000	\$f20=00000000	\$f28=00000000
\$f5=00000000	\$f13=00000000	\$f21=00000000	\$f29=00000000
\$f6=00000000	\$f14=00000000	\$f22=00000000	\$f30=00000000
\$f7=00000000	\$f15=00000000	\$f23=00000000	\$f31=00000000

Segmento de datos

MEMORIA

[0x10010000] 0x616c6f48 0x6e756d20 0x00006f64

PILA

[0x70000000]

