

Clase 12 - RISC - V - Parte 2

Profesor: **IIC2343 - Arquitectura de Computadores**

- Felipe Valenzuela González

Correo:

frvalenzuela@alumni.uc.cl

Resumen de la clase pasada

RISC - V

- Tiene 32 registros de propósito general, cada uno de 32 bits.
- Direcciones de memoria de 32 bits. Las palabras de memoria son de 32 bits (4 bytes), pero **las direcciones son por cada byte**
- Las instrucciones de operaciones aritméticas/lógicas tienen la estructura: palabra clave, registro de destino
- El **registro de destino** es donde se va a almacenar el valor de la operación.

RISC - V

Registro	Nombre	Uso
x0	zero	Registro zero. Siempre almacena el valor 0, no se puede escribir.
x1	ra	Return Address. Se usa para guardar la dirección de retorno de una subrutina.
x2	sp	Stack Pointer. Almacena la dirección del último elemento del <i>stack</i> .
x3	gp	Global Pointer.
x4	tp	Thread Pointer.
x5-x7	t0 – t2	Registros temporales. Pueden perder su valor al llamar una subrutina.
x8-x9	s0 – s1	Registros guardados. Preservan su valor al llamar una subrutina.
x10-x11	a0 – a1	Argumentos de funciones / Valores de retorno.
x12-x17	a2 – a7	Argumentos de funciones.
x18-x27	s2 – s11	Registros guardados. Preservan su valor al llamar una subrutina.
x28-x31	t3 – t6	Registros temporales. Pueden perder su valor al llamar una subrutina.

¿Dudas?

RISC - V

- Tiene 32 registros de propósito general, cada uno de 32 bits.
- Direcciones de memoria de 32 bits. Las palabras de memoria son de 32 bits (4 bytes), pero **las direcciones son por cada byte**
- Las instrucciones de operaciones aritméticas/lógicas tienen la estructura: palabra clave, registro de destino
- El **registro de destino** es donde se va a almacenar el valor de la operación.

RISC - V

Registro	Nombre	Uso
x0	zero	Registro zero. Siempre almacena el valor 0, no se puede escribir.
x1	ra	Return Address. Se usa para guardar la dirección de retorno de una subrutina.
x2	sp	Stack Pointer. Almacena la dirección del último elemento del <i>stack</i> .
x3	gp	Global Pointer.
x4	tp	Thread Pointer.
x5-x7	t0 – t2	Registros temporales. Pueden perder su valor al llamar una subrutina.
x8-x9	s0 – s1	Registros guardados. Preservan su valor al llamar una subrutina.
x10-x11	a0 – a1	Argumentos de funciones / Valores de retorno.
x12-x17	a2 – a7	Argumentos de funciones.
x18-x27	s2 – s11	Registros guardados. Preservan su valor al llamar una subrutina.
x28-x31	t3 – t6	Registros temporales. Pueden perder su valor al llamar una subrutina.

¿Dudas?

La sección .data

- Aquí se definen variables y arreglos.
- Solo usaremos .byte y .word.
- .byte:
 - 1 byte
 - valores pequeños (0–255)
 - byte 10

- .word

4 bytes

enteros normales

.word 123

```
1 a: .byte 10      # ocupa 1 byte
2 b: .word 5000    # ocupa 4 bytes
3 nums: .word 1,2,3,4
4
5
```

La sección .data

- Aquí se definen variables y arreglos.
- Solo usaremos .byte y .word.
- .byte:
 - 1 byte
 - valores pequeños (0–255)
 - byte 10
- .word
 - 4 bytes
 - enteros normales
 - .word 123

```
1 a: .byte 10      # ocupa 1 byte
2 b: .word 5000    # ocupa 4 bytes
3 nums: .word 1,2,3,4
4
5
```

La sección **.text**

- Aquí va el código ejecutable.
- Comienza con la etiqueta `main`:
- Cada línea es una instrucción del procesador.
- Trabajamos con **registros**, no con variables directas.

Instrucción ecall

- Sirve para llamar al sistema y hacer acciones como imprimir números o terminar el programa.
- Lo que hace depende del valor en a7

```
1  lw a0, res      # valor a imprimir
2  addi a7, zero, 7 # código 7 = imprimir entero
3  ecall
4
5  addi a7, zero, 10 # código 10 = salir
6  ecall
7
```

¿Dudas?

Instrucción lw

- Carga un número desde memoria a un registro.
- Formato:

lw destino, dirección

```
1  lw t0, x      # carga el valor guardado en "x"
2  lw t1, 0(t0)  # carga desde la
3                  # dirección apuntada por t0
4
5
6
7
```

Instrucción sw

- Guarda un número desde un registro hacia memoria.
- Formato:

sw registro, dirección

```
1 sw t0, res # guarda el valor de t0 en la variable "res"
2 .
3 .
4 .
5 .
```

Instrucción beq (branch if equal)

- Salta a una etiqueta si dos registros son iguales.
- Se usa para crear bucles o condiciones.
- Formato:

beq reg1, reg2, etiqueta

```
1  
2 beq t1, zero, fin    # si t1 == 0, salta a "fin"  
3  
4  
5
```


¿Dudas?

EJERCICIOS !!

Clase 12 - RISC - V - Parte 2

Profesor: **IIC2343 - Arquitectura de Computadores**

- Felipe Valenzuela González

Correo:

frvalenzuela@alumni.uc.cl