

# Arquitectura de Computadoras 2008

---

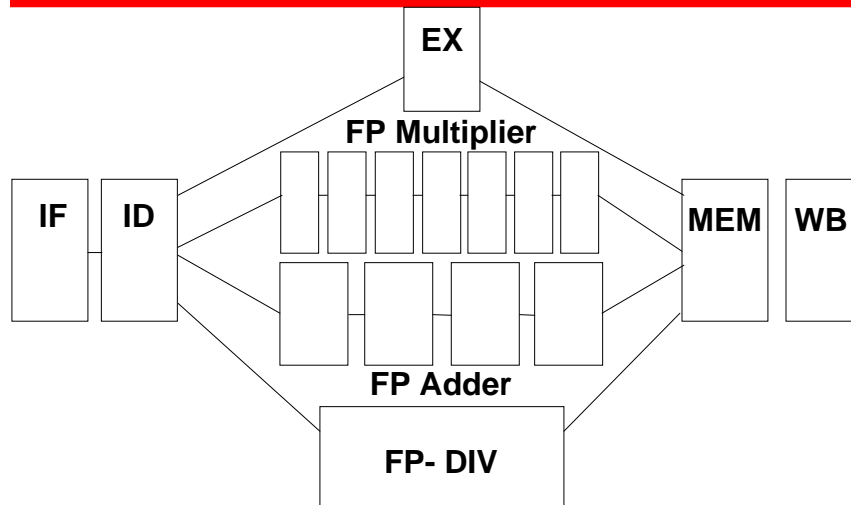
## Simulador WINMIPS64

### Procesador MIPS

---

- 32 registros de uso general: r0 .. r31 (64 bits)
  - excepto r0 siempre igual a 0
- 32 registros de punto flotante: f0 .. f31 (64 bits)
- $2^{30}$  palabras de memoria (32 bits c/u)
- Instrucciones de 1 palabra de longitud (32 bits)
- Acceso a memoria limitado a 2 instrucciones
  - LOAD (carga de memoria en un registro)
  - STORE (almacena un registro en memoria)

## Segmentación en el MIPS



MIPS

Arquitectura de Computadoras 2008

3

## Segmentación en MIPS



### ⌘ Búsqueda (IF)

- ☑ Se accede a memoria por la instrucción
- ☑ Se incrementa el PC

### ⌘ Decodificación / Búsqueda de operandos (ID)

- ☑ Se decodifica la instrucción
- ☑ Se accede al banco de registros por los operandos
- ☑ Se calcula el valor del operando inmediato con extensión de signo (si hace falta)
- ☑ Si es un salto, se calcula el destino y si se toma o no

### ⌘ Ejecución / Dirección efectiva (EX)

- ☑ Si es una instrucción de proceso, se ejecuta en la ALU
- ☑ Si es un acceso a memoria, se calcula la dirección efectiva
- ☑ Si es un salto, se almacena el nuevo PC

### ⌘ Acceso a memoria / terminación del salto (MEM)

- ☑ Si es un acceso a memoria, se accede

### ⌘ Almacenamiento (WB)

- ☑ Se almacena el resultado (si lo hay) en el banco de registros

MIPS

Arquitectura de Computadoras 2008

4

## Directivas al assembler (MIPS64)

---

- .data** - comienzo de segmento de datos
- .text** - comienzo de segmento de código
- .code** - comienzo de segmento de código (= .text)
- .org <n>** - dirección de comienzo
- .space <n>** - deja n bytes vacios
- .asciiz <s>** - entra string ascii terminado en cero
- .ascii <s>** - entra string ascii

donde <n> es un número como 24 y <s> denota un string como "fred".

## Directivas al assembler (2)

---

- .word <n1>,<n2>..** - entra word(s) de dato (64-bits)
- .byte <n1>,<n2>..** - entra bytes
- .word32 <n1>,<n2>..** - entra número(s) de 32 bit
- .word16 <n1>,<n2>..** - entra número(s) de 16 bit
- .double <n1>,<n2>..** - entra número(s) en floating-point

donde <n1>,<n2>.. son números separados por comas.

## E/S del MIPS64

---

E/S mapeada en memoria.

Dirección de CONTROL= 0x10000 y DATA=0x10008

Si CONTROL = 1, Set DATA con Entero S/S para sacar

Si CONTROL = 2, Set DATA con Entero C/S para sacar

Si CONTROL = 3, Set DATA con Punto Flotante para sacar

Si CONTROL = 4, Set DATA con dirección comienzo de string para sacar

Si CONTROL = 5, Set DATA+5 con coordenada X, DATA+4 con coordenada Y  
y DATA con color RGB para sacar

Si CONTROL = 6, limpia la pantalla terminal

Si CONTROL = 7, limpia la pantalla gráfica

Si CONTROL = 8, leer DATA (sea un entero o pto fte) del teclado

Si CONTROL = 9, leer un byte de DATA, sin eco de caracter.

MIPS

Arquitectura de Computadoras 2008

7

## Instrucciones Load/store

---

- LD R1, offset(R2) ; Load Doubleword (64 bits)
  - LB R1, offset(R2) ; Load Byte
    - LBU R1, offset(R2) ; Load Byte s/signo
  - LH R1, offset(R2) ; Load Halfword (16 bits)
    - LHU R1, offset(R2) ; Load Halfword s/signo
  - LW R1, offset(R2) ; Load Word (32 bits)
    - LWU R1, offset(R2) ; Load Word s/signo
- SD R1, offset(R2) ; Store Doubleword
  - SB R1, offset(R2) ; Store Byte
  - SH R1, offset(R2) ; Store Halfword
  - SW R1, offset(R2) ; Store Word

MIPS

Arquitectura de Computadoras 2008

8

## Instrucciones ALU inmediatas

---

- DADDI R1,R2,7 ; R1= R2 + Inmediato
  - DADDUI R1,R2,7 ; R1= R2 + Inmediato s/signo
- SLTI R1,R2,7 ; si R2<Inmediato then R1=1
- ANDI R1,R2,7 ; R1= R2 And Inmediato
- ORI R1,R2,7 ; R1= R2 Or Inmediato
- XORI R1,R2,7 ; Exclusive Or Immediate

MIPS

Arquitectura de Computadoras 2008

9

## Instrucciones ALU en registros

---

- DADD R1,R2,R3 ; R1= R2 Add R3
  - DADDU R1,R2,R3 ; R1= R2 Add R3 s/signo
- DSUB R1,R2,R3 ; R1= R2 Subtract R3
  - DSUBU R1,R2,R3 ; R1= R2 Subtract R3 s/signo
- SLT R1,R2,R3 ; Si R2<R3 then R1=1
- AND R1,R2,R3 ; R1= R2 And R3
- OR R1,R2,R3 ; R1= R2 Or R3
- XOR R1,R2,R3 ; R1= R2 Exclusive Or R3

MIPS

Arquitectura de Computadoras 2008

10

## Instrucciones ALU en registros

---

### de desplazamiento:

- DSLL R1,R2,4 ; Shift Left Logical
  - DSLLV R1,R2,R3 ; idem anterior Variable
- DSRL R1,R2,4 ; Shift Right Logical
  - DSRLV R1,R2,R3 ; idem anterior Variable
- DSRA R1,R2,4 ; Shift Right Arithmetic
  - DSRAV R1,R2,R3 ; idem anterior Variable

## Instrucciones Punto Flotante

---

### Movimiento:

- L.D F1, offset(R0) ; Load Double precision float
- S.D F1, offset(R0) ; Store Double precision float
- MTC1 F1, R1 ; Move Word a Floating Point
- MOV.D F1, F2 ; Move Floating Point

## Instrucciones Punto Flotante (2)

---

### Aritméticas:

- ADD.D F1, F2, F3 ; Floating Point Add
- DIV.D F1, F2, F3 ; Floating Point Divide
- MUL.D F1, F2, F3 ; Floating Point Multiply
- SUB.D F1, F2, F3 ; Floating Point Subtract

## Instrucciones Punto Flotante (3)

---

### Conversión:

- CVT.L.D F1,F2 ; Floating Point a entero  
; (64bits)
- CVT.W.D F1,F2 ; Floating Point a entero  
; (32bits)

## Instrucciones de control de flujo

---

### Salto incondicional:

- J offset ; Jump a offset
- JAL offset ; Jump and Link a offset
- JR R1 ; Jump a dir. en Registro

## Instrucciones de control ... (2)

---

### Salto condicional que compara 2 registros:

- BEQ R1, R2, offset ; si  $R1 = R2$  saltar a offset
  - BNE R1, R2, offset ; si  $R1 \neq R2$  saltar a offset

### Salto condicional que compara con cero:

- BEQZ R1, offset ; si  $R1 = 0$  saltar a offset
  - BNEZ R1, offset ; si  $R1 \neq 0$  saltar a offset



## Otras instrucciones

- NOP ; No Operación
- HALT ; Detiene el Simulator

## Formato Instrucciones Tipo-R

aritmético-lógicas

Op.	Rs	Rt	Rd	Shamnt	funct
6	5	5	5	5	6

Ej: DADD R8, R17, R18       $R8 = R17 + R18$

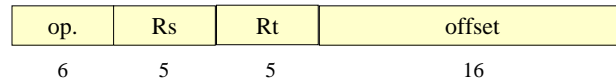
op	17	18	8	0	funct
6	5	5	5	5	6

Ej: SLT R1, R2, R3      if  $R2 < R3$  then  $R1=1$  else  $R1=0$

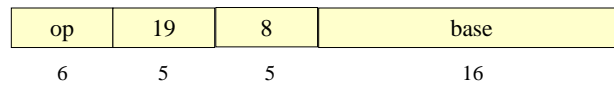
op	2	3	1	0	funct
6	5	5	5	5	6

## Formato Instrucciones Tipo-I

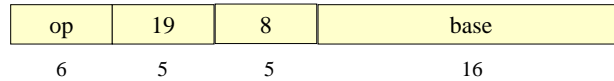
### inmediatas



Ej: LD R8, base (R19)       $R8 = M[\text{base} + R19]$



Ej: SD R8, base (R19)       $M[\text{base} + R19] = R8$



MIPS

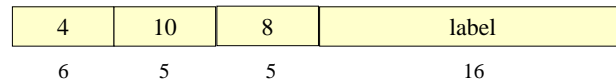
Arquitectura de Computadoras 2008

19

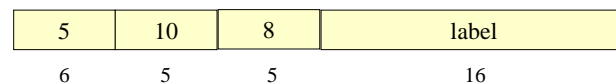
## Formato Instrucciones Tipo-I (2)

### ramificación o salto condicional

Ej: BEQ R8, R10, label      if  $R8 = R10$  goto label



Ej: BNE R8, R10, label      if  $R8 \neq R10$  goto label



MIPS

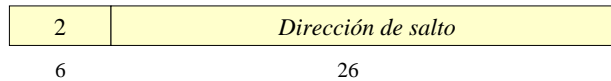
Arquitectura de Computadoras 2008

20

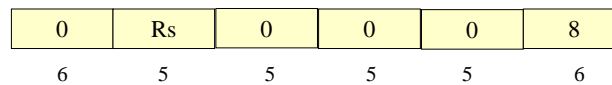
## Formato Instrucciones de control

### instrucciones de salto

Ej: J *dir-de-salto*      PC = *dir-de-salto*



Ej: JR R3      PC = R3



## Llamadas a procedimientos

EL MIPS no tiene pila de hardware, almacena la dirección de retorno **siempre** en R31

- SALTO A SUBROUTINA (Jump And Link)

JAL *dir-de-salto*      R31 = PC  
J *dir-de-salto*

- RETORNO DE SUBROUTINA

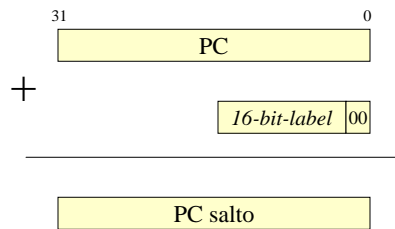
JR R31      PC = R31

## Comparación de los saltos

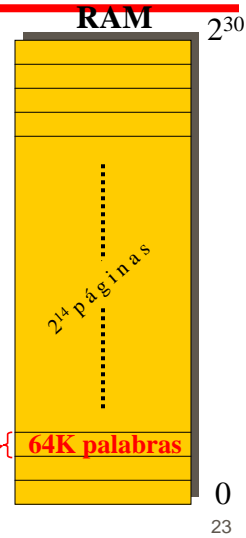
- **BEQ**: salto corto dentro de página

BEQ R4, R5, 16-bit-label

**If (R4==R5) PC = PC + (16-bit-label << 2)**



Rango de salto



MIPS

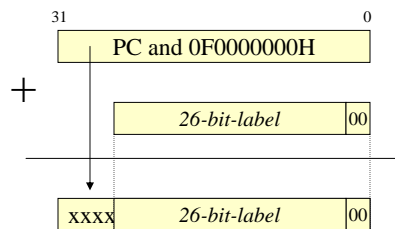
Arquitectura de Computadoras 2008

## Comparación de los saltos (2)

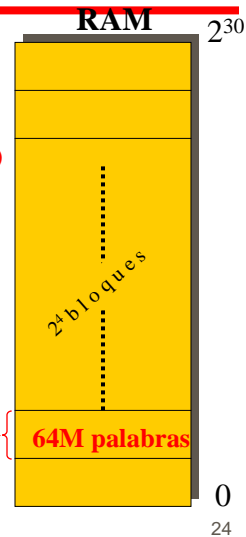
- **J**: salto largo dentro de bloque

J 26-bit-label

**PC = (PC and 0F0000000H) + (26-bit-label << 2)**



Rango de salto



MIPS

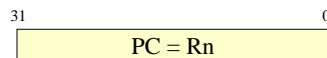
Arquitectura de Computadoras 2008

## Comparación de los saltos (3)

- **JR**: salto largo a toda la memoria

JR  $R_n$

**PC =  $R_n$**



Rango de salto

**RAM**

$2^{30}$

**1G palabras**

0

MIPS

Arquitectura de Computadoras 2008

25

## Ejemplo 1

; C=A+B

.data

A: .word 10

B: .word 8

C: .word 0

.text

main: ld r4, A(r0) ; A en r4

ld r5, B(r0) ; B en r5

dadd r3, r4, r5 ; r3 = r4+r5

sd r3, C(r0) ; resultado en C

halt

MIPS

Arquitectura de Computadoras 2008

26

## Ejemplo 2

for i =1 to 1000 do  
A[i]:=B[i]+5;

```
.data
base_B: .word 1,2,3,4,5,6, ...,1000
base_A: .space 1000
.text
DADDI R2, R0, 1      ; variable I = 1 (en R2)
DADDI R5, R0, 5       ; R5 = 5
DADDI R10, R0, 1001   ; límite del FOR (en R10)
ciclo: LD      R1, base_B(R2) ; R1 = B[I]
DADD  R1, R1, R5      ; R1 = B[I] + 5
SD    R1, base_A(R2)  ; A[I] = R1
DADDI R2, R2, 1       ; I = I + 1
BNE   R2, R10, ciclo  ; I <> 1001 => ir a ciclo
HALT
```

MIPS

Arquitectura de Computadoras 2008

27

## Ejemplo 3

```
.data
busca: .word 7
vect: .word 1,4,8,10,7
largo: .word 5
```

```
.text
dadd R10,R0,R0 ; registro R10 puesto en '0'
dadd R1,R0,R0 ; registro R1 elegido como indice
ld   R2,largo(R0) ; calculamos la dimension del vector vect.
dssl R2,R2,3 ; multiplico R2 x 8
ld   R3,busca(R0) ; elemento buscado
loop: ld   R4,vect(R1) ; elemento del vector a comparar
      beq  R3,R4,found ; salgo de loop si son iguales
      daddi R1,R1,8 ; R1++ (8 byte)
      slt  R5,R1,R2 ; comparo (resultado en R5)
      bnez R5,loop ; continuo el ciclo?
      j    end ; el valor buscado no se encontró
found: daddi R10,R0,1 ; coloco TRUE en R10
end:    halt ; comando winmips de cierre
```

MIPS

Arquitectura de Computadoras 2008

28