MIPS

Microprocessor without Interlocked Pipeline Stages

Surgió a comienzos de los 80 en Stanford.

Sintetiza las principales ideas de RISC.

Arquitectura eficiente y simple.

Procesador MIPS - Registros

32 registros de 32 bits de propósito general (GPR).

32 registros de 32 bits de punto flotante (FPR).

R0 siempre tiene el valor 0.

Instrucciones diferenciadas para GPR y FPR.

Convenciones de registros en MIPS

Name	Register number	Usage	Preserved on call?
$zero	0	The constant value 0	n.a.
$v0–$v1	2–3	Values for results and expression evaluation	no
$a0–$a3	4–7	Arguments	no
$t0–$t7	8–15	Temporaries	no
$s0–$s7	16–23	Saved	yes
$t8–$t9	24–25	More temporaries	no
$gp	28	Global pointer	yes
$sp	29	Stack pointer	yes
$fp	30	Frame pointer	yes
$ra	31	Return address	yes

MIPS - tipos de datos

Bytes de 8 bits.

Half word - Media palabra 16 bits.

Word - Palabra de 32 bits.

Double Word - Doble palabra de 64 bits.

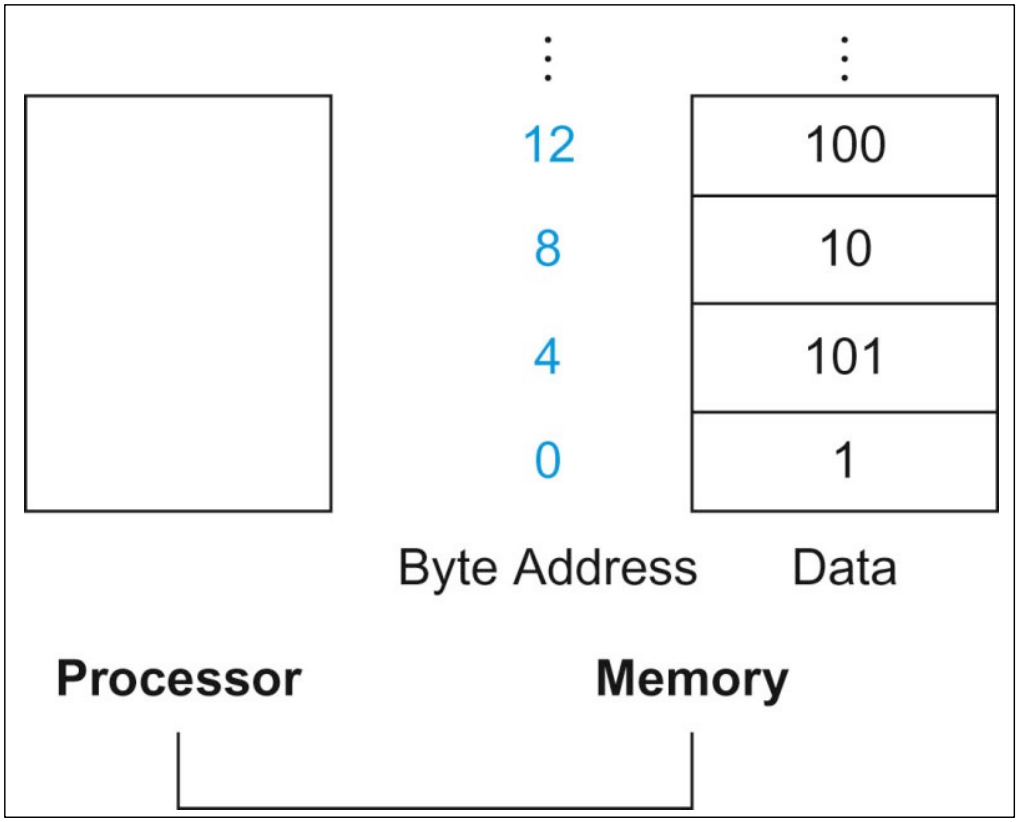
Simple precisión 32 bits.

Doble precisión 64 bits.

} punto flotante

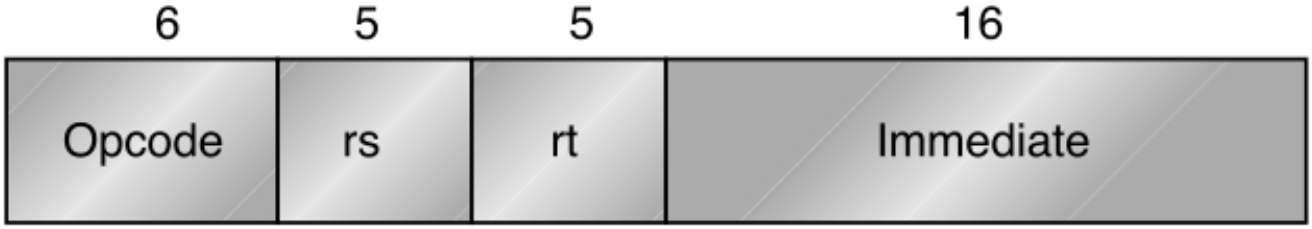
Los bytes, half-word y word son cargados en GPR's, completando con 0 o el signo.

Direcciones de memoria



Formato de instrucciones

I-type instruction



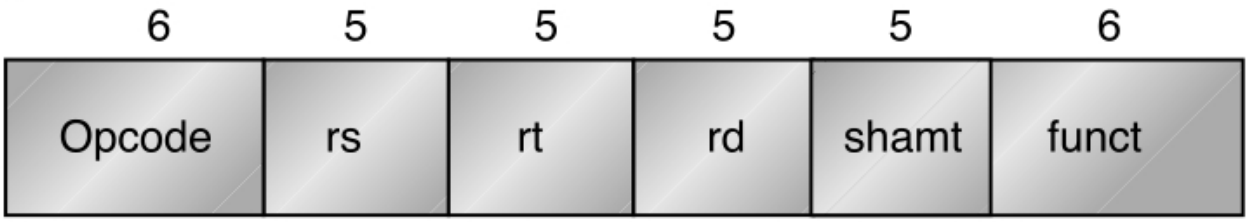
Encodes: Loads and stores of bytes, half words, words, double words. All immediates (rt ← rs op immediate)

Conditional branch instructions (rs is register, rd unused)

En el código de operación se codifica el tipo de dato

Formato de instrucciones

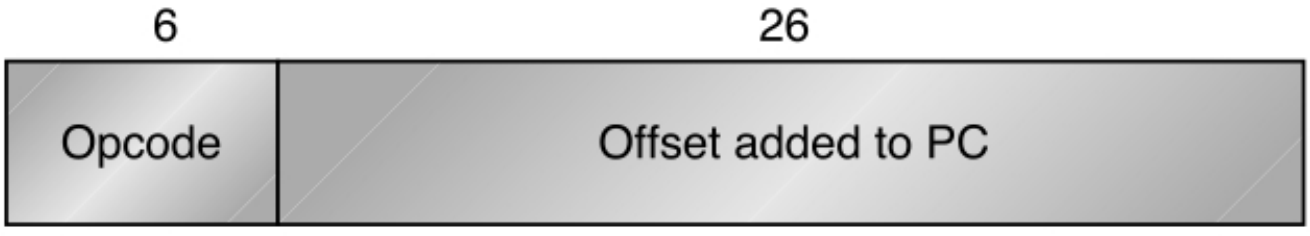
R-type instruction



Register-register ALU operations: rd ← rs funct rt
Function encodes the data path operation: Add, Sub, . . .
Read/write special registers and moves

Formato de instrucciones

J-type instruction

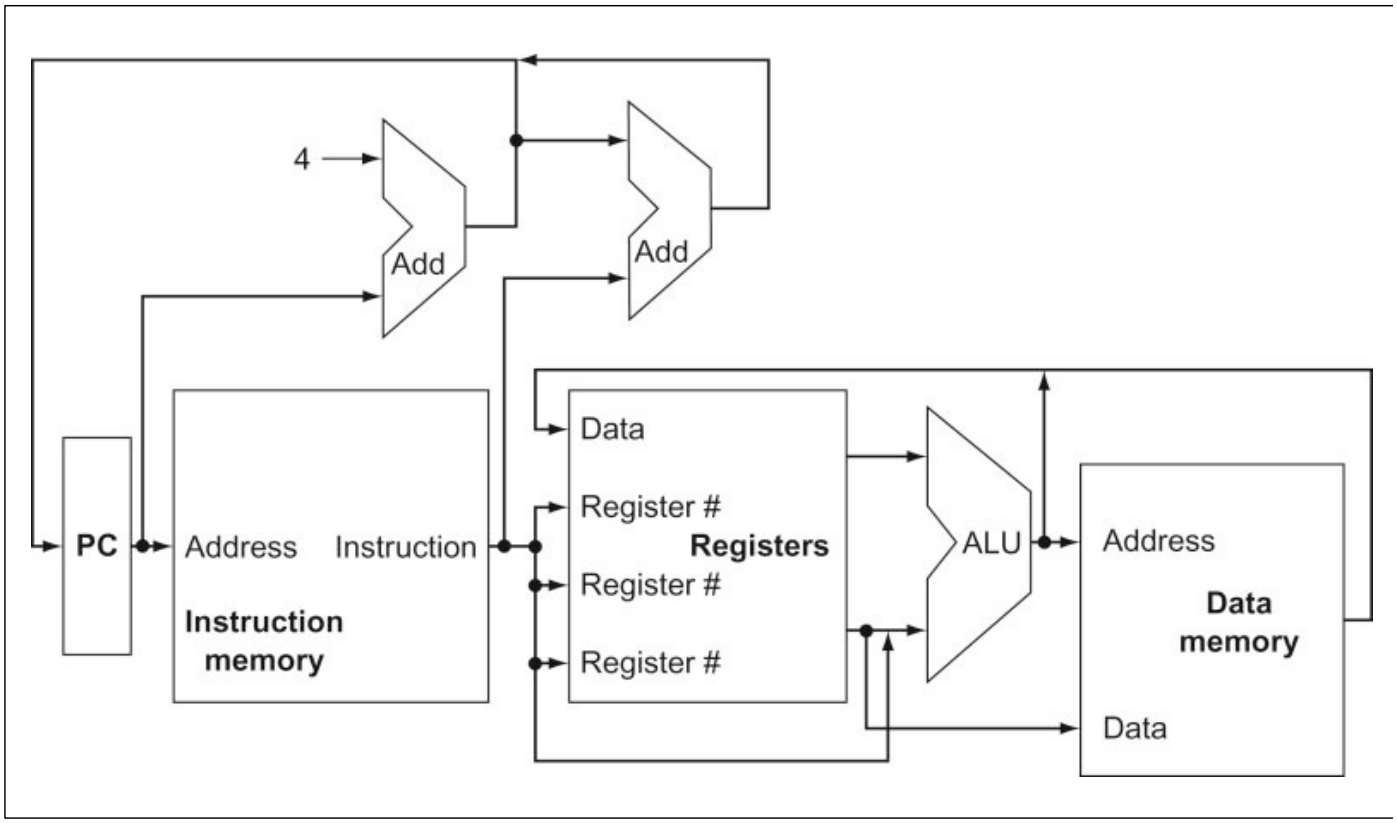


Jump and jump and link
Trap and return from exception

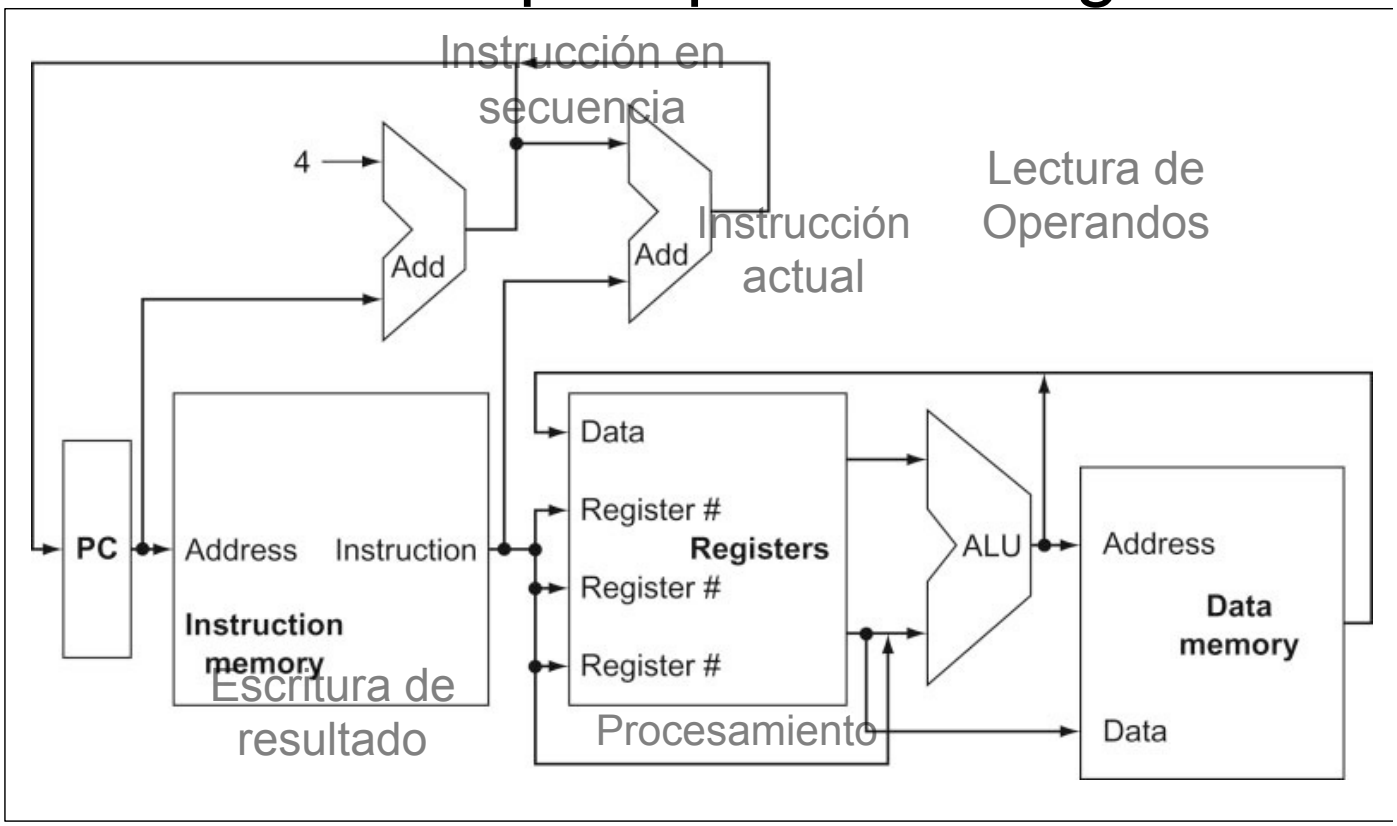
Los 3 formatos de instrucciones

Name	Fields					
Field size	6 bits	5 bits	5 bits	5 bits	5 bits	6 bits
R-format	op	rs	rt	rd	shamt	funct
I-format	op	rs	rt	address/immediate		
J-format	op	target address				

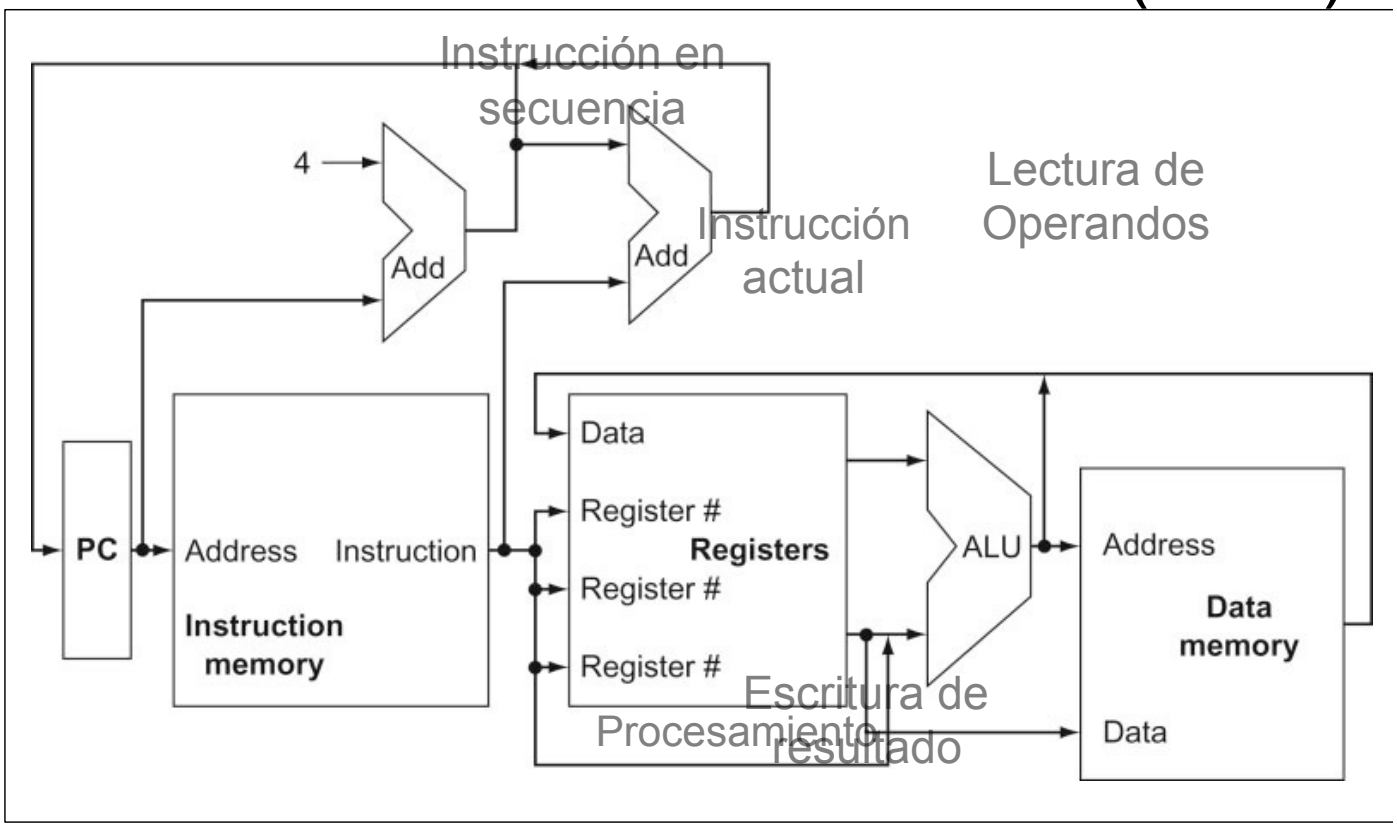
Visión abstracta de un MIPS



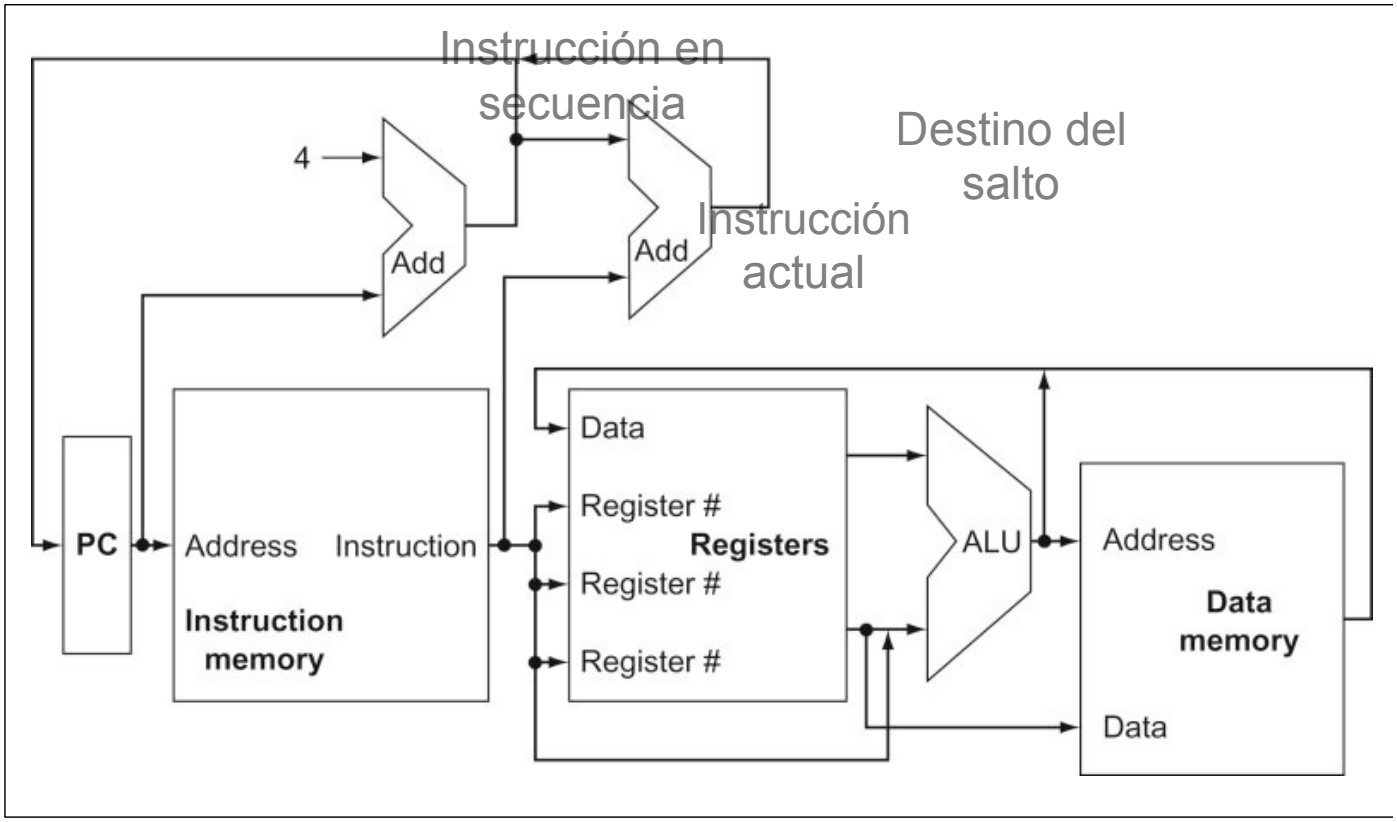
Instrucción que opera con registros



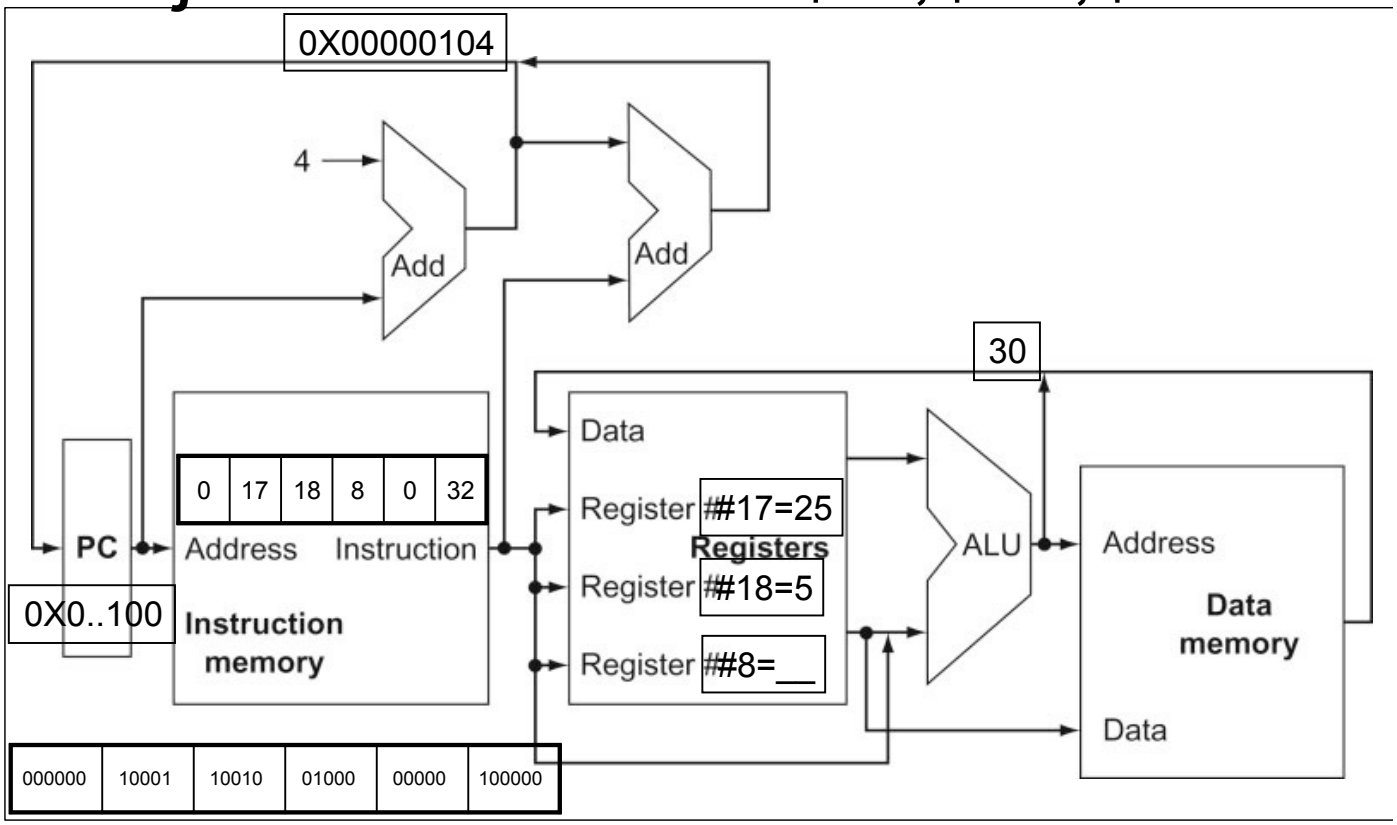
Instrucción de almacenamiento(store)



Instrucción de salto Incondicional



Ejecución de add $t0,$s1,$s2



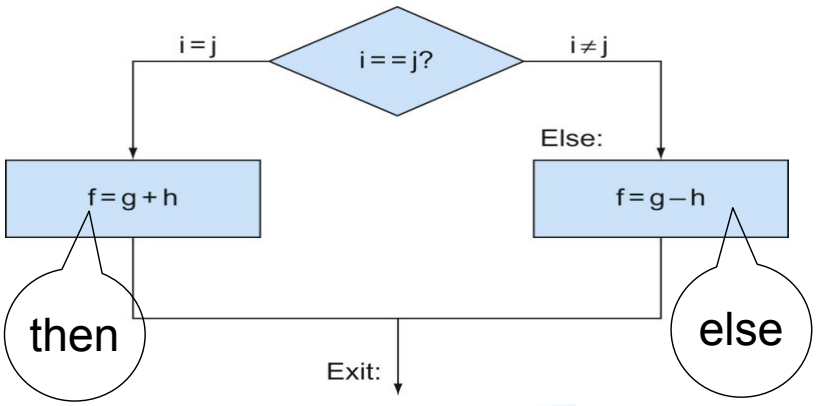
Instrucciones de salto condicional

beq reg1, reg2, rótulo1 (Branch if equal)

ir a la sentencia en rótulo1, si (reg1)=(reg2)

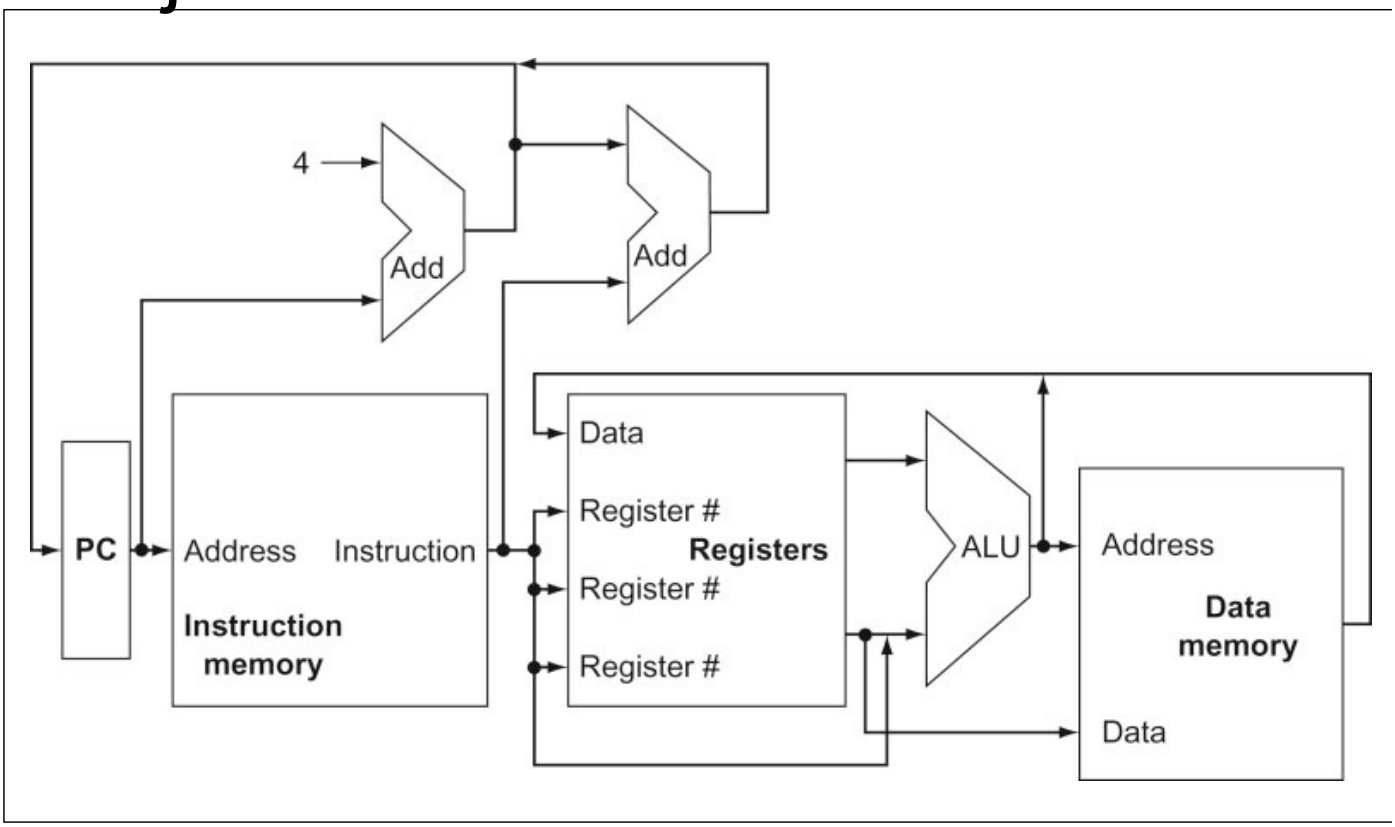
bne reg1, reg2, rótulo1 (Branch if not equal)

ir a la sentencia en rótulo1, si (reg1)≠(reg2)



if(i==j) f=g+h; else f=g-h;

Ejecución de salto condicional



MIPS - Ejemplo de Operaciones

lw $s0,30($t0) load word

Regs[16] $\leftarrow_{32}$ Mem[30+Regs[8]]

sll $a0,$t2,5 Shift izquierda lógico

Regs[4] $\leftarrow_{32}$ Regs[10]<<5

Operaciones de control de secuencia

Instrucciones de comparación. slt $t1,$t2,$t3

Si (Regs[t2]<Regs[t3])

Regs[t1] ←1 Else Regs[t1] ←0

jump	jump and link	jump register
PC ← Off<<2 : PC	$ra=PC+4 PC ← Off<<2	PC ← Regs[rs]

26 bits

Los Branches son condicionales. Testean dos registros fuente. PC ← Inm<<2 + PC

16 bits

Invocación a procedimientos

Poner los parámetros donde el procedimiento pueda accederlos

Transferir el control al procedimiento.

Adquirir los recursos de memoria que necesita el procedimiento

Realizar la tarea deseada.

Poner el resultado donde el programa pueda accederlo.

Retornar el control al punto original.

Convenciones en MIPS
$a0-$a3 Registros para cuatro argumentos
$v0,$v1 Registros para retornar valores
$ra Registro para la dirección de retorno. jal $ra

Uso de la pila (lifo)

Resgurdo de registros del llamador.

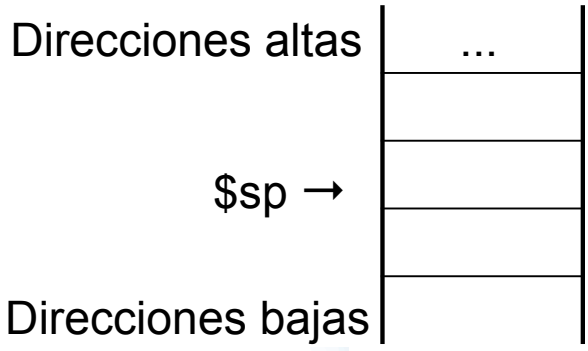
El puntero al tope de la pila es $sp (29)

La pila crece hacia las direcciones bajas.

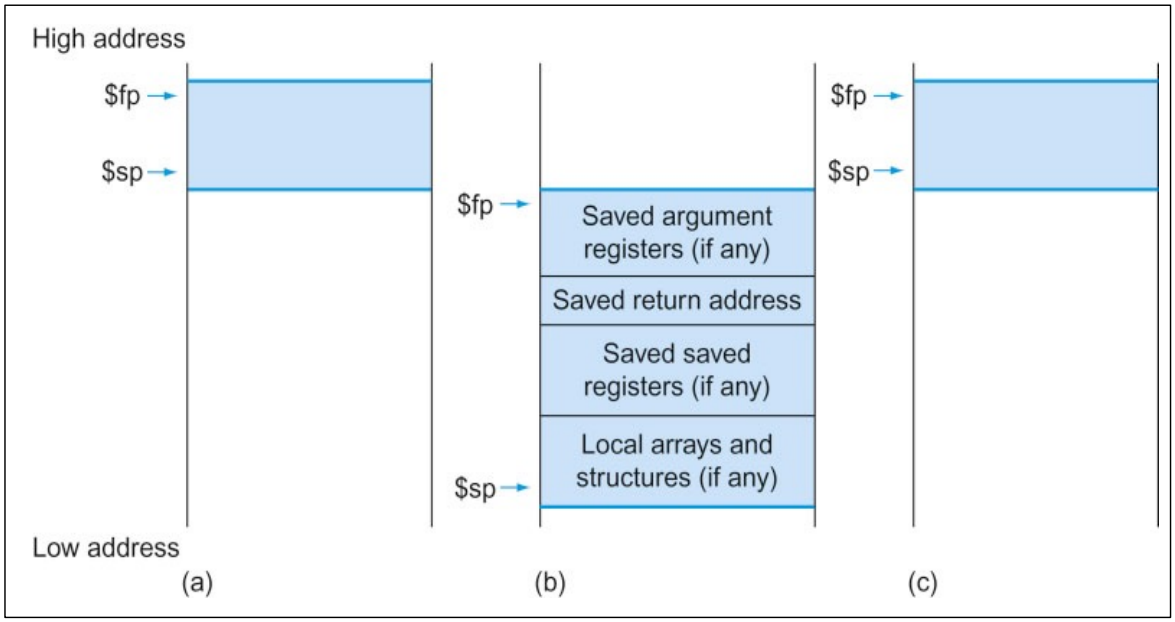
$t0-$t9 no preservados por el procedimiento

$s0-$s7 preservados.

Anidamiento de llamadas y variables locales



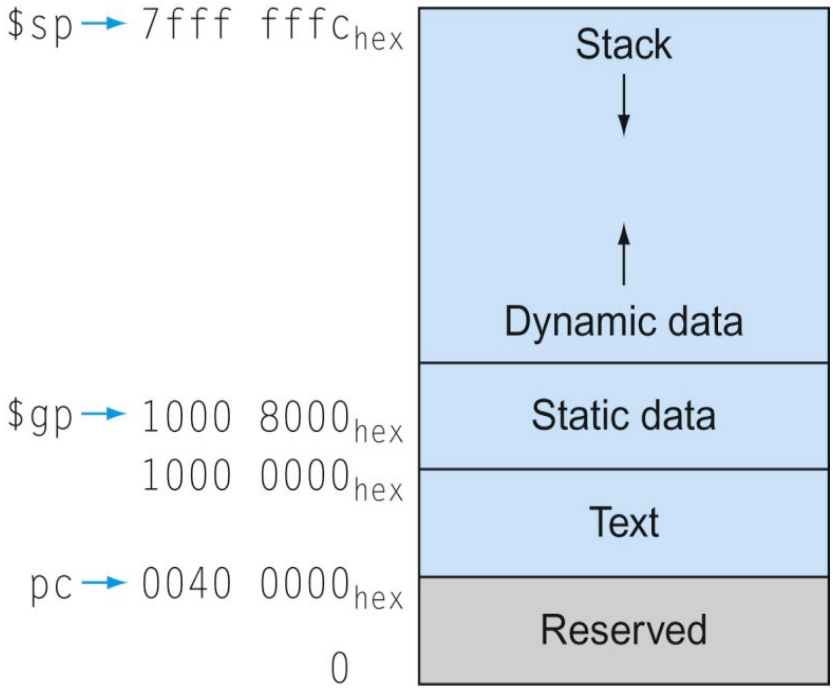
Registro de activación



Puntero al marco $fp (30)

Base estable para las referencias locales

Memoria para código y datos



MIPS - modos de direccionamiento

Inmediato. Campo de 16 bits.

addi $t2,$t3,64

Desplazamiento. Campo de 16 bits.

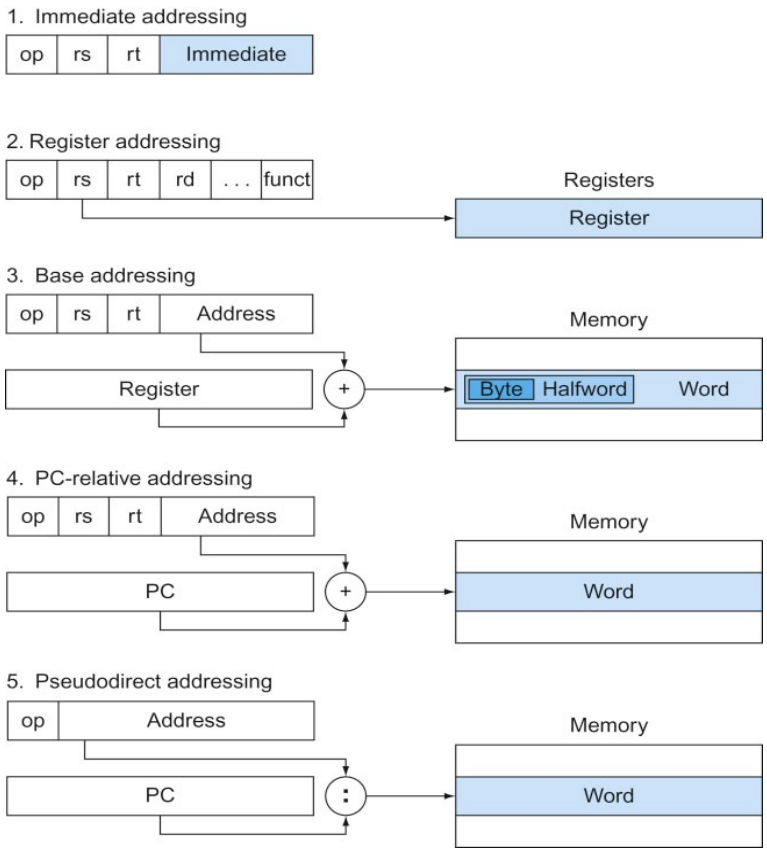
load $s1,8($s3)

Direcciona a byte. Tamaño de dirección 32 bits.

Transferencia desde y hacia memoria con load y store.

Acceso a datos alineados.

Modos de direccionamiento



Direcciones e inmediatos de 32 bits

The machine language version of `lui $t0, 255  # $t0 is register 8:`

001111	00000	01000	0000 0000 1111 1111

Contents of register `$t0` after executing `lui $t0, 255:`

0000 0000 1111 1111	0000 0000 0000 0000

j 0x10000 #ir a la dirección 0x10000	
2	0x4000<<2
6 bits	26 bits

bne $s0,$s1,Exit # ir a Exit si $s0≠$s1			
5	16	17	Exit
6 bits	5 bits	5 bits	16 bits

Pseudoinstrucción	Traducción
mov $rt, $rs	addi $rt, $rs, 0
li $rs, small	addi $rt, $rs, small
li $rs, big	lui $rs, upper(big) ori $rs, $rs, lower(big)
la $rs, big	lui $rs, upper(big) ori $rs, $rs, lower(big)

lw $rt, big($rs)	lui $t0, upper(big)
	ori $t0, $t0, lower(big)
	add $t0, $rs, $t0
	lw $rt, 0($t0)

```
        #Recorre un arreglo de palabras hasta
        #que encuentra una distinta a cero
                .text
main:           la $s6,arr
                add $s3,$zero,$zero
                add $s5,$zero,$zero
loop:           sll $t1,$s3,2
                add $t1,$t1,$s6
                lw $t0,0($t1)
                bne $t0,$s5,exit
                addi $s3,$s3,1
                j loop
exit:           jr $ra
                .data
arr:            .word 0,0,1
```

base del arreglo

indice *4

dirección del elemento

carga la palabra

incremento del índice

```
while(arr[i]==k)
    i++;
```


¿Qué vimos?

- Características de los procesadores RISC.
  - Instrucciones simples.
  - Direccionamientos sencillos.
  - Ejecución eficiente.
- Procesador MIPS
  - Registros y tipos de datos.
  - Modos de direccionamientos
  - Formatos de instrucciones.
  - Ejemplos de operaciones.