Taller 3:

viernes, 18 de marzo de 2022 11:20

Fundamentos de Computadores

TALLER 3 - REPERT

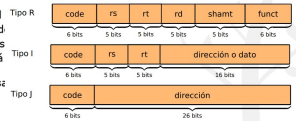
El objetivo del taller es realizar los ejer
de prácticas. El taller se realizará en g
profesor.

El grupo será evaluado en función del
grupo, así como de la presentación d
cada apartado, el grupo (o el profes
presentará los resultados y responderá

Todos los miembros del grupo deben sa

Tres formatos diferentes (de 32 bits):

- **Tipo R:** instrucciones aritmético/lógicas y otras.
- **Tipo I:** Instrucciones de transferencia de datos, saltos. condicionales e instrucciones con operandos inmediatos.
- **Tipo J:** Instrucciones de salto incondicional.



Enunciado del taller

Se ha modificado el repertorio de instrucciones del MIPS de 32 bits de acuerdo con las siguientes características:

- Hay 128 operaciones de 32 bits que permiten hasta 4 operandos (uno destino y tres fuente).
- Se dispone de un sistema de 64 registros de 32 bits.
- Se utilizan los modos de direccionamiento del MIPS vistos en clase.

Dado el siguiente fragmento de código:

addi $4, $5, 4 → Se almacena en $4 la suma de $5+4
add3 $4, $5, $6, $7 → $5+$6+$7 → $4
lw $4, 4($7) → Carga en $4 la palabra almacenada en $7 + 4 (desplazamiento)
beq $4, $3, salto → Compara y salta si son iguales

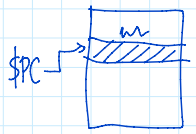
j 2500 → Modifica el valor del PC para ejecutar las instrucciones en 2500

1. Indicar qué hace cada una de las instrucciones del código. Explicar el formato de cada una (campos, número de bits por campo, significado de cada campo). Escribir para cada instrucción el código máquina correspondiente.
2. Calcular las direcciones de memoria menor y mayor a las que se puede referenciar para las instrucciones **lw, beq** y **j** de dicho código. Especificar los resultados en hexadecimal. La dirección de la instrucción **addi** es **0x10a0c304**. → Ej. 7/T.3 es parecido.

* a registro
* Relativo al PC

NOTA: No se puede usar calculadora para la realización y exposición del ejercicio.

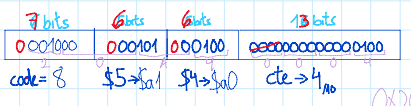
* Pseudodirecto



② addi → 0x10a0c304 —add3→ 0x10a0c308 —lw→ 0x10A0C3C

0001 0000 1010 0000 1100 0011 0000 0100

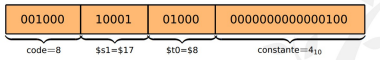
addi $4, $5, 4



0x20A40004

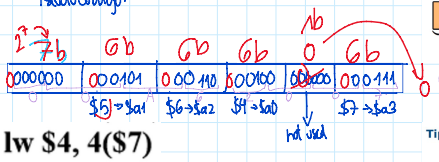
Tipo I con direccionamiento inmediato

- Suma inmediato: `addi $t0,$s1,4` ($t0 ← $s1+4$_{10}$)



add3 $4, $5, $6, $7

Pseudocódigo.



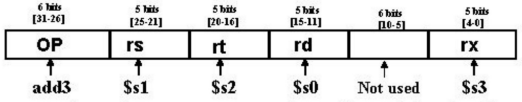
Tipo R con direccionamiento registro

- Suma: `add $t0,$s1,$s2` ($t0 ← $s1+$s2)

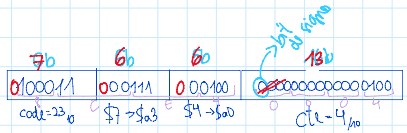
add3 $s0,$s1, $s2, $s3

Register $s0 gets the sum of $s1, $s2 and $s3.

The instruction encoding uses a modified R-format, with an additional register specifier rx added replacing the five low bits of the "funct" field.



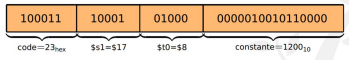
lw $4, 4($7)



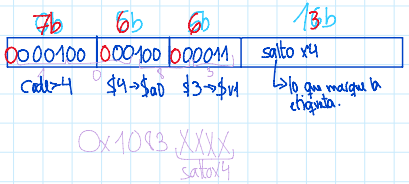
0x8CE46004

Tipo I con direccionamiento base con desplazamiento

- Load word: `lw $t0,1200($s1)` ($t0 ← Mem[$s1+1200$_{10}$])

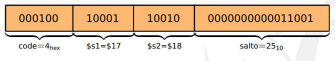


beq $4, $3, salto → Salto condicional



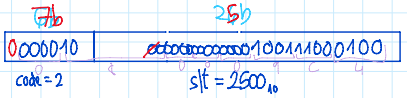
Tipo I con direccionamiento relativo al PC

- Branch on equal: `beq $s1,$s2,25` (if $s1==$s2 goto PC+(25×4))



PC anterior = 0100 1101 0000 0011 1010 1100 0101 1000
\+ 0000 0000 0000 0000 0000 0000 0110 0100
PC nuevo = 0100 1101 0000 0011 1010 1100 1011 1100

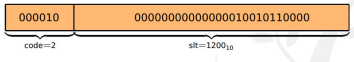
j 2500 → Salto incondicional.



0x08009C4

Tipo J con direccionamiento pseudodirecto

- Jump: `j 1200`



PC anterior = 0100 1101 0000 0011 1010 1100 0101 1000
PC nuevo = 0100 0000 0000 0000 0001 0010 1100 0000
1200₁₀