# MIPS

OPERACIONES ARITMÉTICAS

# MIPS.

- El estudio del ensamblador de una arquitectura permite comprender y dominar el diseño y el funcionamiento de cada una de las partes de la misma.

# Registros

- MIPS dispone, entre otros, de los registros siguientes:
  - 32 Registros de 32b en la CPU.
  - 32 Registros de 32b en la unidad de coma flotante.
  - Contador de programa (PC) de 32b.
  - 2 Registros de 32 b para mult y div
- Normas de buen uso:
  - $s0..$s7, $sp →llamadas a funciones
  - $t0..$t9→modificado en llamadas a funciones

# Operaciones aritméticas básicas

- MIPS es una máquina de arquitectura carga-almacenamiento: Los accesos a memoria se hacen a través de un registro.
- Las operaciones aritméticas básicas en MIPS se caracterizan por:
  - Usar 3 registros (uno destino, dos operandos)
  - Sintaxis: cop resultado op1 op2
  - El último operando puede ser un valor inmediato de 16 bits.

# Operaciones aritméticas básicas

- Ejemplo de una suma:

C

```
int a, b, c;
c = a + b;
```

MIPS

```
# Suponiendo que los datos a, b, c están
# asignados a los registros $s0,$s1,$s2
add $s2, $s0, $s1
```

# Operaciones aritméticas básicas

- Sólo se pueden sumar 2 registros

Código C:

```
int a, b, c, d, e;
a = (b + c) – (d + e);
```

Código MIPS:

```
# Datos están asignados a los registros
$s0,$s1,$s2,$s3,$s4
add $t0, $s1, $s2
add $t1, $s3, $s4
sub $s0, $t0, $t1
```

# Ejercicios

1) Modificar el programa MIPS anterior para evitar el uso de los registros temporales $t0 y $t1.
2) Factorial de 5.
3) 34-9
4) 100/50
5) (4 + 3+ 5) – ((9 * 2) + 8)

# Registros

**Carga inmediata en registros**

• Recordando que el registro $zero siempre vale 0, podemos utilizarlo para asignar valores a los otros registros mediante la instrucción addi (“sumar inmediato”).

Ejemplo:

- addi $s0, $zero, 100 # $s0 = $zero + 100 = 0 + 100 = 100

Esta instrucción sólo nos permite utilizar inmediatos de 16 bits. Al igual que en otros casos que veremos más adelante, se produce extensión de signo del bit 15 del inmediato (bit de signo) del 31 al 16.

Para utilizar los 32 bits de los registros, debemos usar la pseudoinstrucción li:

- li $s0, 0xABCDEF01 # $s0 = ABCDEF0116

Esta pseudoinstrucción equivale a la siguiente secuencia de instrucciones:

- lui $s0, 0xABCD # $s0 = ABCD16 * 216 = ABCD000016
- ori $s0, $s0, 0xEF01# $s0 = $s0 OR EF01 = ABCD000016 OR EF0116 = # = ABCDEF0116