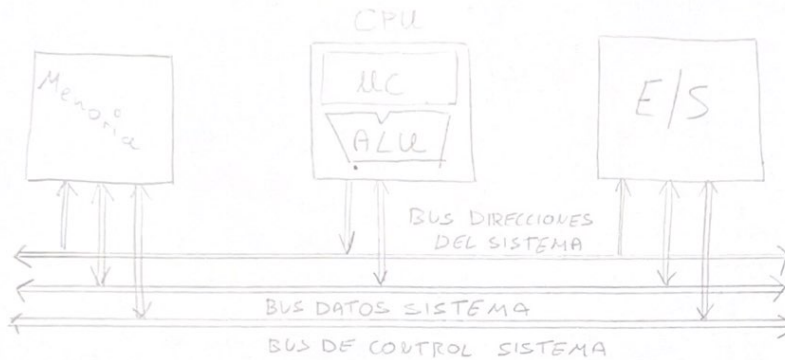


TEMA 1: INTRODUCCIÓN

Definición de ordenador/computador: máquina capaz de recibir información de entrada, procesarla bajo el control de un programa y generar información a través de medios de salida, sin intervención humana.



Arquitectura  
Von Neumann

PARTES DE LA ARQUITECTURA

ALU  $\equiv$  Arithmetic Logic Unit: Parte de la CPU encargada de realizar las operaciones aritméticas y lógicas.

CU  $\equiv$  Control Unit: Su función es, buscar las instrucciones del programa en la memoria principal, interpretarlas y ejecutarlas bajo la FU (Functional Unit), encargada de realizar los cálculos y operaciones llamados por los programas.

Bus de direcciones: A través de este bus, la CPU indica a la memoria o a los periféricos E/S donde se encuentra la información a leer o escribir.

Bus de datos: Sobre este bus se transmite la información entre los diferentes elementos del computador.

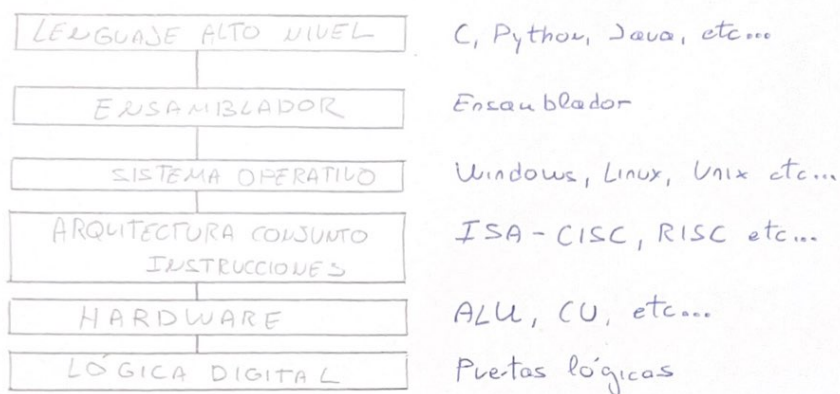
Bus de control: Contiene líneas encargadas de gestionar la transmisión de información en el computador.

Técnicas para la ejecución de programas en lenguaje máquina:

Traducción: Antes de ejecutar el programa L1, hay que traducirlo a las instrucciones en lenguaje L0.

Interpretación: El programa L1 se envía como entrada a un programa L0 llamado intérprete que se encarga de ejecutarlo sin necesidad de traducirlo a L0.

### NIVELES DE ABSTRACCIÓN



MIPS64  $\equiv$  Millones de Instrucciones por Segundo

RISC  $\equiv$  Reduced Instruction Set Computing  $\equiv$  instrucciones simples y más rendimiento

CISC  $\equiv$  Complex Instruction Set Computing  $\equiv$  instrucciones complejas y facilidad programación

### RENDIMIENTO

Métrica: magnitud de cuantifica un aspecto medible de un sistema.

Tiempo de respuesta  $\equiv T_r$ : Tiempo que invierte el computador en realizar una tarea.

$$T_r = t_{fin} - t_{llegada}$$

$$\text{Productividad} = \frac{\text{Tareas completadas}}{\text{Tiempo de referencia}}$$

Productividad: n° tareas completadas por unidad tiempo

### Formas de Mejorar el rendimiento de procesador

1° Hay que cambiar el procesador si o si

→ Dos opciones

→ Procesador con doble frecuencia

→ Procesador con más núcleos (2)

→ En el primer caso las tareas tardan  $t/2$  en ejecutarse, por tanto la productividad se duplica.

→ Al no modificar el sistema inicial para cada uno de los núcleos, las tareas siguen teniendo tiempo  $t$ , pero la productividad se duplica.



02-09-2022

## ARQUITECTURA DE COMPUTADORES

Aceleración  $\equiv$  ratio/diferencia que existe entre el rendimiento de un ordenador y otro.

$$\text{Aceleración} = \frac{\text{Rendimiento}_A}{\text{Rendimiento}_B} = \frac{\text{Productividad}_A}{\text{Productividad}_B}$$

La Aceleración puede ser  $\begin{cases} = 1, \text{ mismo rendimiento} \\ < 1, B \text{ tiene mejor rendimiento que } A \\ > 1, A \text{ tiene mayor rendimiento que } B \end{cases}$

Latencia = tiempo que tarda el sistema en proporcionar los datos solicitados

Ancho de banda  $\equiv$  Cantidad de información por unidad de tiempo

### LEY DE AMDAHL

$$T_{\text{respuesta mejorado}} = T_{\text{respuesta original}} \cdot \left( \left( 1 - \text{Fracción mejorada} \right) + \frac{\text{Fracción mejorada}}{\text{Aceleración mejorada}} \right)$$

Deducción Andahl  $\rightarrow A = \frac{\text{Rendimiento mejorado}}{\text{Rendimiento original}} = \frac{T_{\text{respuesta original}}}{T_{\text{respuesta mejorado}}}$

POR TANTO 
$$\text{Andahl} = \frac{1}{\left( 1 - \text{Fracción mejorada} \right) + \frac{\text{Fracción mejorada}}{\text{Aceleración mejorada}}}$$

$\text{CPI} \equiv \text{Ciclos por Instrucción} \Rightarrow \text{CPI} = \frac{\text{Ciclos CPU Programa}}{(N) \text{ Instrucciones del Programa}} *$

RECUERDA  $f = \frac{1}{T}$

$T_{\text{tiempo CPU}} = \frac{\text{Ciclos}^* \text{ CPU Programa}}{\uparrow} \Rightarrow T_{\text{CPU}} = \text{CPI} \cdot \text{Instrucciones Programa} \cdot T$   
ESTO SE LLAMA LEY DE HIERRO

(N) Instrucciones de Programa  $\rightarrow$  Tipo de arquitectura

$\text{CPI} \rightarrow$  Organización interna + complejidad

$T \rightarrow$  Periodo  $\rightarrow$  frecuencia CPU

Benchmark: Programa pensado para evaluar el rendimiento de un computador o partes

FIU TEMA 1

$\rightarrow$  Dos tipos

$\left\{ \begin{array}{l} \text{Carga real: Programas reales, rendimiento real, difícil de reproducir.} \\ \text{Sintética: Programas pequeños, condiciones reales, resultados pueden diferir.} \end{array} \right.$

## TEMA 2: LA CPU

24/10/2022

### Arquitectura MIPS64:

Es una arquitectura RISC basado en el modelo de máquina de carga/almacenamiento. Es una extensión de MIPS32, con la que es totalmente compatible.

MIPS64 tiene 32 registros desde R0 hasta R31 para enteros de 64 bits. también hay que tener en cuenta que R0 ~~R4~~ siempre vale 0 y R31 está reservado para instrucciones de salto, concretamente para almacenar su dirección.

Además existen otros 32 registros de punto flotante, FPO...FP31

### Tipos de datos:

Byte: es un número entero de 8 bits.  $(2^3)$

HalfWord: entero de 16 bits (2 Bytes)  $(2^3 \cdot 2)$

Word: entero de 32 bits (4 Bytes)  $(2^3 \cdot 2 \cdot 2)$

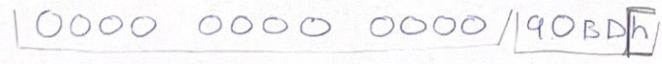
DWord: entero de 64 bits (8 bytes)  $(2^3 \cdot 2 \cdot 2 \cdot 2)$

### Almacenamiento con/su signo

Cuando se produce la carga de un valor de menos de 64 bits (Byte, Halfword, word) se puede realizar con o sin signo.


#### Sin signo

Todo el registro se pone a 0 y después se pone el dato en la parte más baja del registro.

Se pone todo a 0  Se añade en la parte más baja

#### Con signo

Primero se copia el dato en la parte más baja del registro y después se replica el bit más significativo en los bits más significativos no modificados

 INDICA QUE ES HEXADECIMAL

0 1111... 111 1 0... 1 0



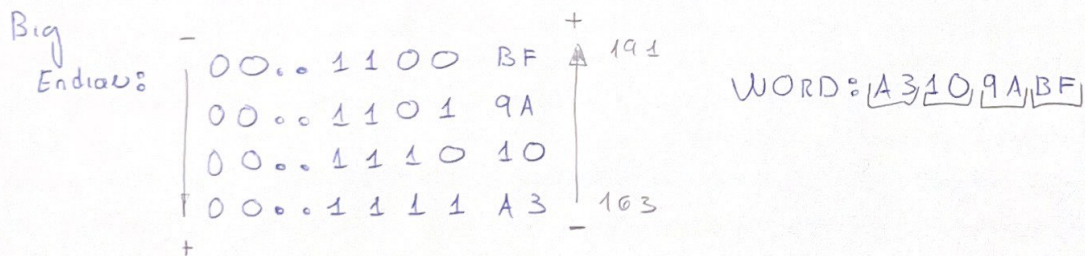
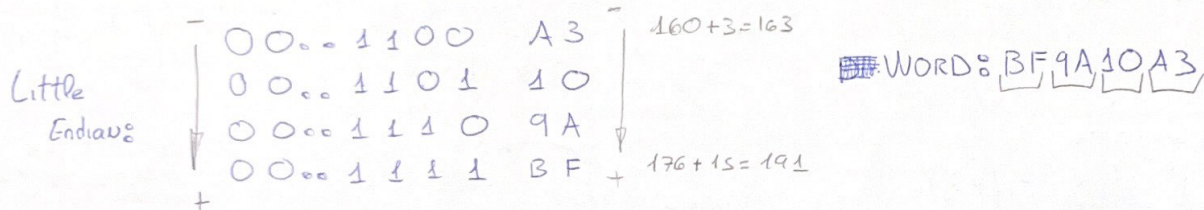
Accesos a memoria con tipos de datos

Todos los accesos de más de 1 byte de memoria deben estar alineados, la posición inicial debe ser múltiplo de 2, 4 u 8 dependiendo del tipo de dato.

Ordenación al almacenar en memoria

Big-endian: Se almacenan los bytes más significativos en la dirección más pequeña, y los menos significativos en la más grande.

Little-endian: Se almacenan los bytes menos significativos en la dirección más pequeña y los más significativos en la más grande.

Operandos y modos de direccionamiento:

Operandos: elemento sobre el que se aplica una operación. (Variable, dato)

Los operandos pueden estar en

- MEMORIA  $\equiv$  Tiempo acceso más elevado
- REGISTROS CPU  $\equiv$  n° de registros escaso
- CÓDIGO INSTRUCCIÓN  $\equiv$  Solo ctes, NO almacenar resultado como valor inmediato.

Existen 5 modos de direccionamiento:

\* Inmediato: El valor del operando va en el código

daddi r4, r8, -3  
DOUBLE WORD      Single Immediate       $r4 \leftarrow r8 + (-3)$

\* Registro: Se incluye el n° de registro dentro del código

mov rd, rs       $rd \leftarrow rs$   
Mover reemplazar

\* Indexado por base: El operando está en el valor de una cte en un registro

ld r12, 200(r0)  
Load Double       $0 + 200$   
r0 = siempre vale 0

$r12 \leftarrow 200(r0)$   
Siempre que se use r0 se habla de direccionamiento directo ya que r0 siempre es 0.  
Carga en r12 la duord en 200(r0) o sea posición 200.

ld r12, 0(r3)  
Load Double      r3

$r12 \leftarrow 0(r3)$   
Siempre que se use 0 como cte se habla de direccionamiento indirecto, la dir es el registro base.  
Carga en r12 la duord en r3

26/10/2022

\* Relativo al

PC (Program counter): Sirve para calcular la dirección de salto cuando el PC y una cte

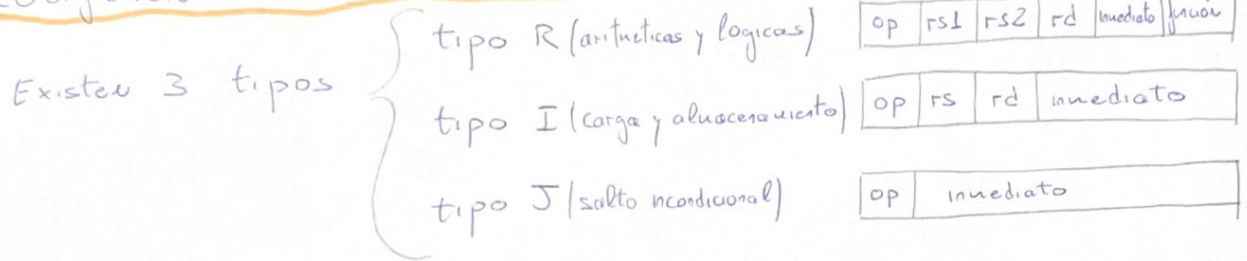
beq r2, r9, -5  
Salto condicional      n° instrucciones a saltar  
Solo si AMBOS registros son iguales

\* Pseudo directo: Sirve para calcular saltos incondicionales

j 1000



29/10/2022

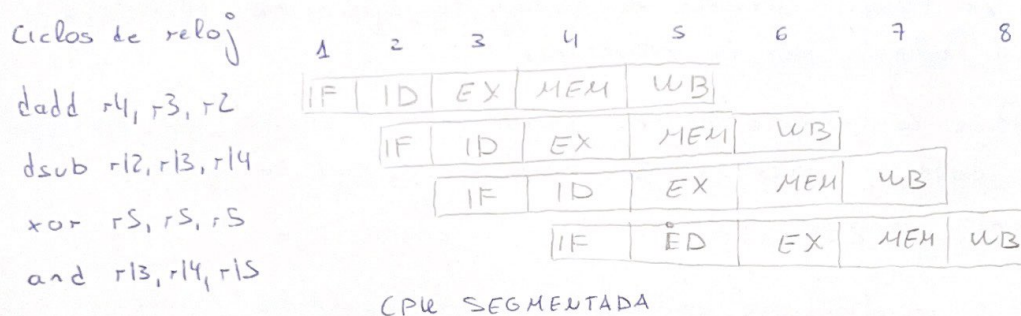
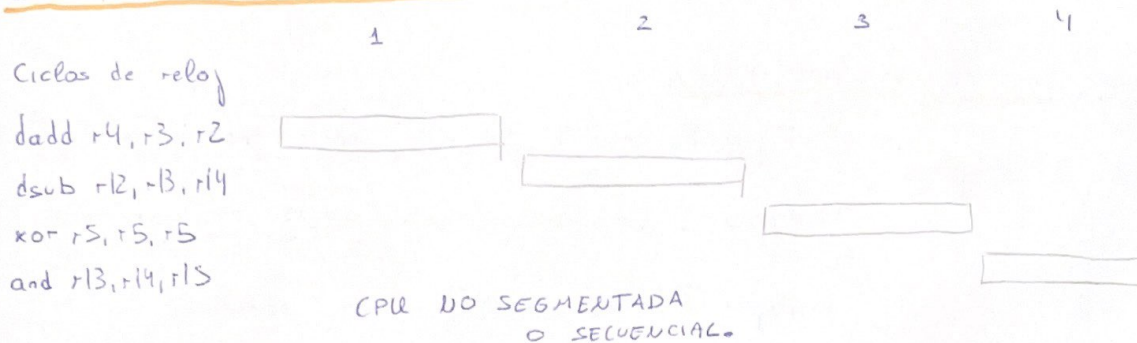
Codificación de instrucciones

- \* Op: Existe en todos, representa el código de operación, identifica la operación realizada por la instrucción.
- \* Función: Indica la variante de la operación del campo op, ejemplo: el op de add y sub es 0, pero su código de función es diferente, 2Ch y 2Eh.
- \* rs/rs1: n° de registro que contiene el primer operando fuente.
- \* rs2: n° de registro que contiene el segundo operando fuente.
- \* rd: n° de registro donde se almacenará el operando destino.
- \* inmediato: Valores inmediatos codificados junto con el resto de código.

Etapas de ejecución de instrucciones

1. IF  $\equiv$  Instruction Fetch: Se busca el código de instrucción en la memoria, se lee desde la memoria de instrucciones el código apuntado por el PC (program counter) y se incrementa 4 posiciones para apuntar a la siguiente instrucción.
2. ID  $\equiv$  Instruction Decode: Se decodifica la instrucción y se buscan los registros operandos, se actualiza el PC con la dirección destino de saltos incondicionales.
3. EX  $\equiv$  Execute: Ejecución del código en la ALU.
4. MEM  $\equiv$  Memory: Las op de carga y descarga acceden a memoria, se actualiza el PC de saltos condicional calculado anteriormente.
5. WB  $\equiv$  Write Back: Se escribe en fichero registros.

## CPU NO SEGMENTADA/SEGMENTADA



## RIESGOS DE SEGMENTACIÓN

Estructurales: Cuando el hardware no es capaz de ejecutar en paralelo una determinada combinación de instrucciones que requieren del mismo recurso.

Datos: Cuando una instrucción referencia un dato (ya sea un registro o posición de memoria) de una instrucción anterior. Hay diferentes tipos de dependencias de datos:

\*RAW: Aparece cuando una instrucción lee un registro previamente escrito por la anterior instrucción. (Read After Write)

sd r3, 120(r0)

ld r5, 120(r0)

— No hay detención ya que en operandos en memoria solo se accede a memoria en la etapa MEM de ins de carga y almacenamiento.

## OJO A OPERANDOS EN REGISTROS

↳ ID no se completa (etapa de decode y lectura) hasta el WB



## ARQUITECTURA DE COMPUTADORES

\* WAR: Sucede cuando antes de la etapa ID de lectura sucede una etapa WB que modifique el valor del registro.

dsub r1, r10, r2    IF ID EX MEM WB  
dadd r2, r3, r4    IF ID EX MEM WB

Si la etapa WB de dadd sucediese antes que ID dsub habría WAR

WAR = WRITE AFTER READ

\* Si se implementase ejecución fuera de orden podríamos encontrar este tipo de dependencia.

\* WAW: Sucede cuando dos instrucciones quieren escribir y la instrucción actual no puede porque la anterior aún no ha escrito.

dadd r1, r10, r2    IF ID EX MEM WB  
and r1, r3, r4    IF ID EX MEM WB

Si el WB de and se fuese a ejecutar antes que el de dadd habría WAW

WAW =  
WRITE AFTER WRITE

\* Similar a WAR, es necesario ejecución fuera de orden y latencia en instrucciones para que se genere esta dependencia.

## RIESGOS DE CONTROL

Ocurren cuando hay cambios en el flujo de ejecución, tales como llamadas a funciones, excepciones, interrupciones o saltos. Es cuando se intenta tomar una decisión sobre una condición aún no evaluada.

## PLANIFICAR INSTRUCCIONES

Es una técnica que implica reordenar las instrucciones de forma que las dependencias de datos no produzcan excepciones.

dadd r2, r3, r4    IF ID EX MEM WB  
dsub r1, r10, r2    IF ID ID ID EX MEM WB  
or r12, r11, r14    IF IF IF ID EX MEM WB

↓  
Dependencias RAW de r2 entre dadd y dsub

30/10/2022

Solución: reorganizar las instrucciones sin alterar la semántica del programa.

dadd r2, r3, r4	IF	ID	EX	MEM	WB		
or r12, r11, r14		IF	ID	EX	MEM	WB	
dsub r1, r10, r2			IF	ID	EX	MEM	WB

Al cambiar or ya no hay dependencia RAW entre dadd y dsub

## RUTAS DE REENVÍO

Conexiones en el camino de datos para reducir las dependencias de tipo RAW, hay dos tipos:

EX  $\rightarrow$  EX      MEM  $\rightarrow$  EX

dadd r2, r3, r4	IF	ID	EX	MEM	WB				
xor r6, r11, r10		IF	ID	EX	MEM	WB			
dsub r1, r10, r2			IF	ID	ID	EX	MEM	WB	
or r12, r11, r14				IF	ID	ID	EX	MEM	WB

\* Sin rutas de reenvío

dadd r2, r3, r4	IF	ID	EX	MEM	WB			
xor r6, r11, r10		IF	ID	EX	MEM	WB		
dsub r1, r10, r2			IF	ID	EX	MEM	WB	
or r12, r11, r14				IF	ID	EX	MEM	WB

\* Con rutas de reenvío, salida mem  $\rightarrow$  entrada ex

## RENOMBRADO DE REGISTROS

Para eliminar las dependencias de datos WAR y WAW existe la posibilidad de renombrar registros en vez de reciclarlos.

ori r5, r0, 1  
ori r6, r0, 3  
dadd r4, r6, r5  
ori r4, r0, 7  
ori r6, r0, 6

$\Rightarrow$  ori r4 no puede comenzar hasta que termine dadd, RAW.  
Pero si ori utilizase otro registro no habría dependencias.

### TIPOS DE REGISTROS

- \* Asociado y en uso: Si contador  $> 0$  y están asociadas a un registro arquitectónico.
- \* Asociado y en desuso: contador  $= 0$  y asociado a un registro, inicialmente r0 - r31 y r32 - r39.
- \* Disponible: contador  $= 0$  y sin asociar, inicialmente son r32 - r63 y r64 - r127.



ori r5, r0, 1	→	ori rr32, rr0, 1
ori r6, r0, 3	→	ori rr33, rr0, 3
dupl (r4), r6, r5	→	ori rr34, rr33, rr32
ori (r4), r0, 7	→	ori rr35, rr0, 7
ori r6, r0, 6	→	ori rr36, rr0, 6