

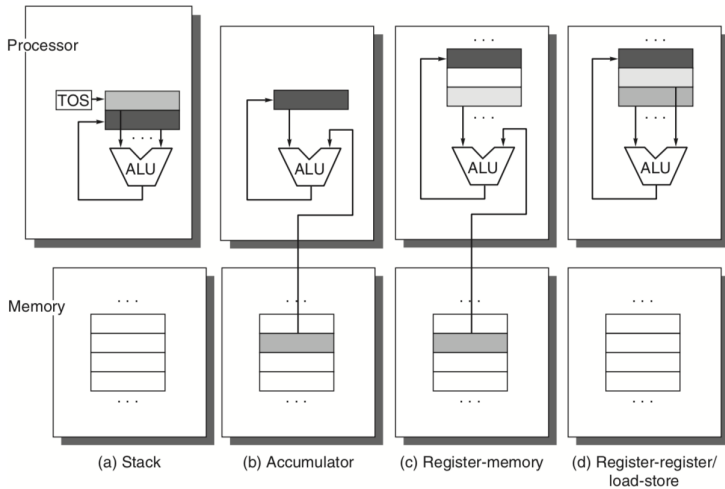
U.B.A. - Facultad de Ingeniería

66.20/86.37 Organización de Computadoras  
Arquitecturas

Práctica

1<sup>er</sup> cuatrimestre 2020

# Clasificación de ISAs



# Clasificación de ISAs

- ▶ Modelo de pila: Los operandos están implícitamente en el tope del stack y el hardware debe evaluar la expresión en un solo orden y cargar un operando múltiples veces.
- ▶ Modelo de acumulador: En una arquitectura de acumulador un operando está implícito en el acumulador.
- ▶ Modelo de registros de propósito general: Aquí se tienen únicamente operandos explícitos, ya sea que estén ubicados en registros o en memoria.
- ▶ Modelo de carga y almacenamiento: En este tipo de arquitectura, la memoria solo puede ser accedida a través de instrucciones de carga y almacenamiento (load/store).

# Tipo de instrucciones en MIPS

- ▶ Load / Store: únicas que acceden a memoria.
- ▶ Computational: realizadas en la ALU.
- ▶ Jump / Branch: saltos incondicionales y condicionales.
- ▶ Miscelaneas: Serialización de instrucciones, Excepciones (syscall), Moves condicionales, Prefetch, NOPs.
- ▶ Coprocessors y FPU.

## Ejercicio 2.1 CAAQA 3ed

Asumir que los valores A, B, C, D y E residen en memoria, que el opcode de la instrucción es de 8 bits, las direcciones de memoria son de 64 bits y las direcciones de registro son de 6 bits.

- a Por cada arquitectura del conjunto de instrucciones mostradas en el primer slide ¿Cuántas direcciones o nombres aparecen en cada instrucción para calcular  $C = A + B$  y cuál es el tamaño total del código?
- b Algunas ISAs de la figura del primer slide destruyen los operando durante el cómputo y trae una consecuencia de performance. Para cada arquitectura escribir la secuencia de código para calcular  $C = A + B$  seguido de  $D = A - E$ . En el código, indicar cada operando que se destruye durante la ejecución e indicar en cada una el “overhead” que se incluye como solución. Dar el tamaño en total del código, los datos que son movidos de o hacia memoria y el overhead de cada secuencia de código.

## Ejercicio 2.2 CAAQA 3ed

Algunas operaciones con dos operando no son conmutativas (la resta por ejemplo). ¿Qué ventajas y desventajas en las arquitecturas de pila, acumulador y carga-almacentamiento existen cuando se ejecuta operaciones no conmutativas?

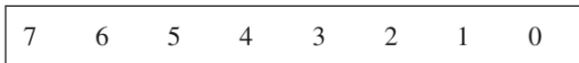
# Endianess

Todo el conjunto de instrucciones es direccionado por bytes y se provee acceso por bytes (8 bits), mitad de palabra (half words) (16 bits), por palabra (words) (32 bits).

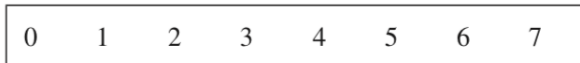
Hay dos convenciones para ordenar los bytes en un objeto. **Little Endian** y **Big Endian**.

# Endianess

El orden **Little Endian** coloca el byte cuya dirección "x . . . x000" en la posición menos significativa. Los bytes entonces son numerados:



El orden **Big Endian** coloca el byte cuya dirección es "x . . . x000" en la posición más significativa. Los bytes entonces son numerados:





# Alineación

En varias computadoras, los accesos a objetos más grandes que un byte deben estar alineados.

- ▶ Un objeto de tamaño  $s$  bytes en la dirección de bytes  $A$  está alineado si  $A \bmod s = 0$ .
- ▶ Esto se fuerza en muchas arquitecturas porque el hardware accede a memoria típicamente alineado a múltiplos de word o double-word.

¿Qué puede ocurrir a nivel de accesos a memoria si se accede a datos no alineados y la arquitectura lo permite?

# Modos de direccionamiento

Se presentan varios modos de direccionamiento:

- ▶ Register
- ▶ Immediate
- ▶ Displacement
- ▶ Register indirect
- ▶ Indexed
- ▶ Direct or Absolute
- ▶ Memory indirect
- ▶ Autoincrement
- ▶ Autodecrement
- ▶ Scaled

MIPS implementa 2 modos de direccionamiento: Immediate y Displacement, ambos con immediate de 16 bits.

Usando el registro 0 se consigue Absolute, y con displacement 0 se consigue Register indirect.

# Modos de direccionamiento

Immediate:	add	\$t1, \$t2, 10
Displacement:	lw	\$t1, 30(\$t2)
Absolute:	sb	\$t1, 40(\$0)
Indirect:	sw	\$t1, 0(\$t2)

## Ejercicio 2.3 CAAQA 3ed

El valor representado por el número hexadecimal 434F 4D50 5554 4552 se almacena en una palabra doble alineada en 64 bits.

- a. Utilizando el arreglo físico de la primera fila en la Figura 2.5, escribir el valor a ser almacenado utilizando el orden de byte Big Endian. Luego, interpretar cada byte como caracter ASCII y debajo de cada byte escribir el caracter correspondiente formando la cadena de caracteres que se almacenaría en el orden Big Endian.
- b. Realizar el punto (a.) pero siguiendo el orden Little Endian.

## Ejercicio 2.3 CAAQA 3ed

Value of 3 low-order bits of byte address									
Width of object	0	1	2	3	4	5	6	7	
1 byte (byte)	Aligned	Aligned	Aligned	Aligned	Aligned	Aligned	Aligned	Aligned	
2 bytes (half word)	Aligned		Aligned		Aligned		Aligned		
2 bytes (half word)		Misaligned		Misaligned		Misaligned		Misaligned	
4 bytes (word)	Aligned				Aligned				
4 bytes (word)		Misaligned				Misaligned			
4 bytes (word)		Misaligned					Misaligned		
4 bytes (word)		Misaligned						Misaligned	
8 bytes (double word)	Aligned								
8 bytes (double word)		Misaligned							
8 bytes (double word)		Misaligned							
8 bytes (double word)		Misaligned							
8 bytes (double word)		Misaligned							
8 bytes (double word)		Misaligned							
8 bytes (double word)		Misaligned							
8 bytes (double word)		Misaligned							

Fiigura 2.5: Direcciones de byte alineadas y no alineadas

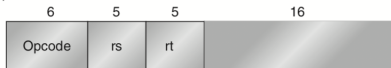
# Encoding MIPS

Hay 3 formatos de encoding posibles en MIPS

- ▶ I-type (Immediate)
- ▶ R-type (Register)
- ▶ J-type (Jump)

# Encoding MIPS

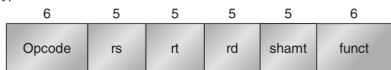
I-type instruction



Encodes: Loads and stores of bytes, half words, words, double words. All immediates ( $rt \leftarrow rs \text{ op immediate}$ )

Conditional branch instructions (rs is register, rd unused)  
Jump register, jump and link register  
( $rd=0$ ,  $rs=\text{destination}$ ,  $\text{immediate}=0$ )

R-type instruction



Register-register ALU operations:  $rd \leftarrow rs \text{ funct } rt$

Function encodes the data path operation: Add, Sub, ...

Read/write special registers and moves

J-type instruction



Jump and jump and link  
Trap and return from exception

# Encoding MIPS

Viendo el detalle de las instrucciones J-type vemos que hay un offset de 26 bits left-shifted 2 bits, para formar los 28 lsb de la direccion destino del salto (las regiones de salto están alineadas a 256MB, y la región está determinada por los 4 msb de PC).

¿Por qué se puede (y conviene) tomar los 26 bits como desplazados a la izquierda 2 bits?



Instrucción	Frecuencia
<i>Load</i>	26%
<i>Store</i>	10%
Aritméticas	14%
Comparaciones	13%
Salto condicionales	16%
Salto incondicionales	1%
<i>Call / Return</i>	2%
Shift	4%
Logicas	4%
Misc.	5%

*Tabla 1: Mix. de instrucciones*

## Ejercicio 2.6 CAAQA 3ed

Varios investigadores han sugerido que incorporar un modo de direccionamiento registro-memoria a una máquina *Load/Store* puede ser conveniente. La idea es reemplazar la siguiente secuencia

```
LOAD  R1, 0(Rb)
ADD   R2, R2, R1
```

por

```
ADD   R2, 0(Rb)
```

Asumir que la nueva instrucción provoca un incremento en el ciclo de clock del 5%. Utilizar la frecuencia de instrucciones de la Tabla 1. La nueva instrucción afecta únicamente al ciclo de clock y no al CPI.

## Ejercicio 2.6 CAAQA 3ed

- a. ¿Qué porcentaje de loads deben ser eliminados en la máquina con la nueva instrucción para tener al menos el mismo rendimiento?
- b. Mostrar una secuencia de instrucciones donde un load de R1 seguido inmediatamente por el uso de R1 (con algún tipo de opcode) no podría ser reemplazado por una única instrucción como la propuesta, asumir que el mismo opcode existe.

## Ejercicio 2.11 CAAQA 3ed

Calcular el CPI Efectivo para MIPS utilizando la Tabla 1. Asumir que se obtuvieron los siguientes CPI para instrucciones.

Instrucción	Ciclos
ALU	1, 0
<i>Load/Store</i>	1, 4
Salto condicional	
Tomados	2, 0
No Tomados	1, 5
Salto	1, 2

Asumir que el 60% de los saltos son tomados.

## Ejercicio 2.12 CAAQA 3ed

Considerar el agregado de un nuevo modo de acceso a MIPS. El nuevo modo suma dos registros y un valor de offset de 11 bits con signo para obtener la dirección efectiva. Utilizar el porcentaje de instrucciones indicado en la Tabla 1. El compilador pasa de las instrucciones:

```
ADD      R1, R1, R2
LW       Rd, 100(R1)    (or Store)
```

A utilizar:

```
LW      Rd, 100(R1,R2)
```

## Ejercicio 2.12 CAAQA 3ed

- a. Asumir que el nuevo modo de acceso es usado por el 10% de los *loads* y *stores*. ¿Cuál es el porcentaje de IC nuevo comparado con la tasa original?
- b. Si el nuevo modo de direccionamiento aumenta en 5% el tiempo de clock, ¿Cuál computadora será más rápida y por cuanto?
- c. Describa una situación que limite el porcentaje de reemplazos realizado.