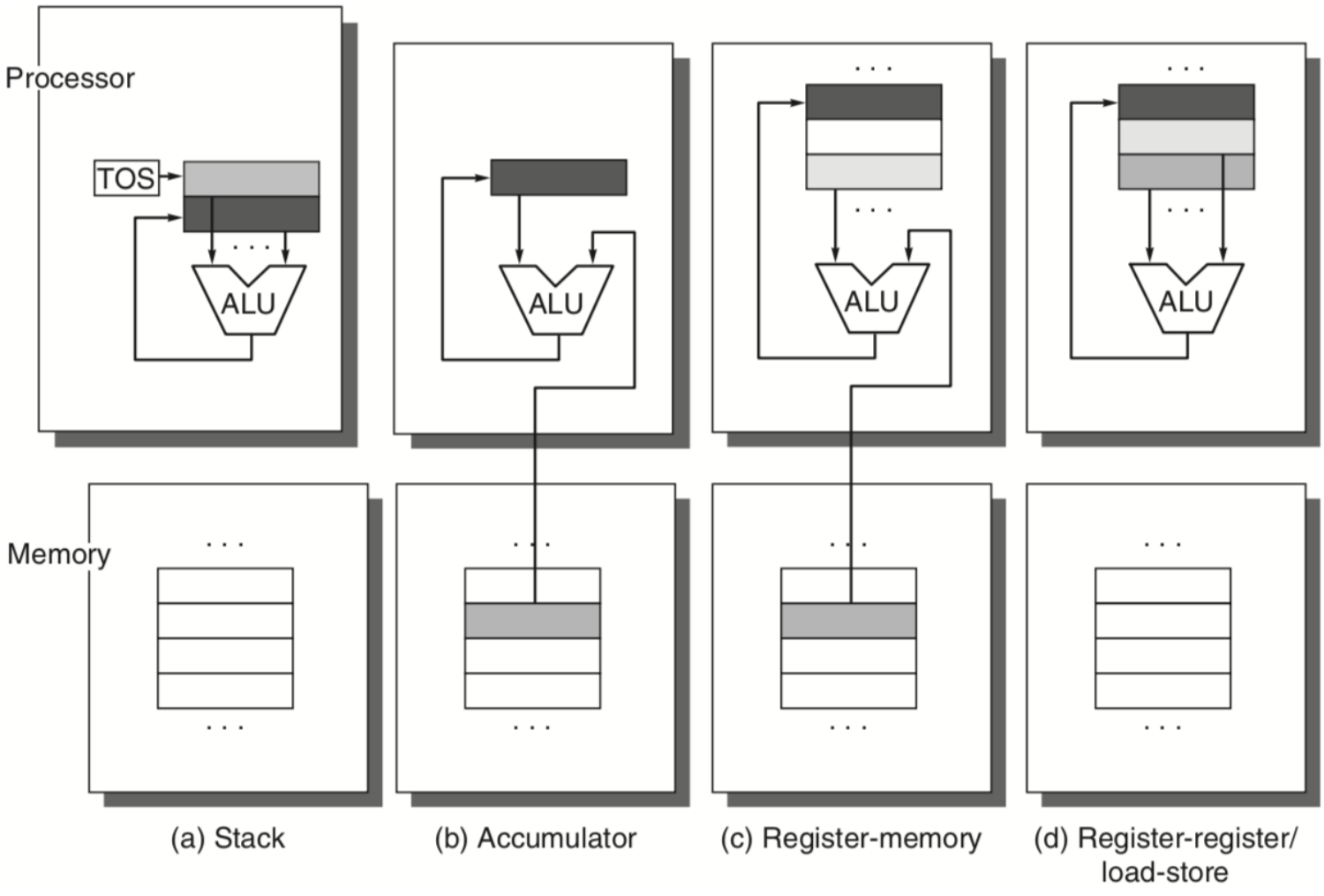
U.B.A. - Facultad de Ingeniería

66.20/86.37 Organización de Computadoras Arquitecturas

Práctica

2^{do} 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.

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 mod 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)
```

Encoding MIPS

Hay 3 formatos de encoding posibles en MIPS

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

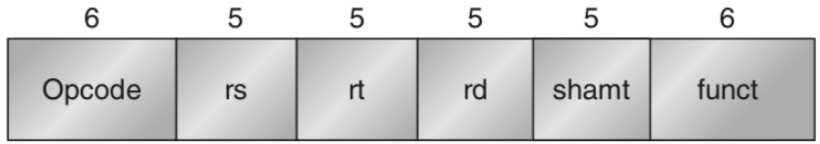
Encoding MIPS

I-type instruction

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

Conditional branch instructions (rs is register, rd unused)
Jump register, jump and link register
(rd=0, rs=destination, immediate=0)

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

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?