Informática

Unidad 1: Introducción

1A: Arquitectura de Computadoras

Ingeniería en Mecatrónica

Facultad de Ingeniería
Universidad Nacional de Cuyo

Dr. Ing. Martín G. Marchetta
mmarchetta@fing.uncu.edu.ar

UNIVERSIDAD NACIONAL DE CUYO

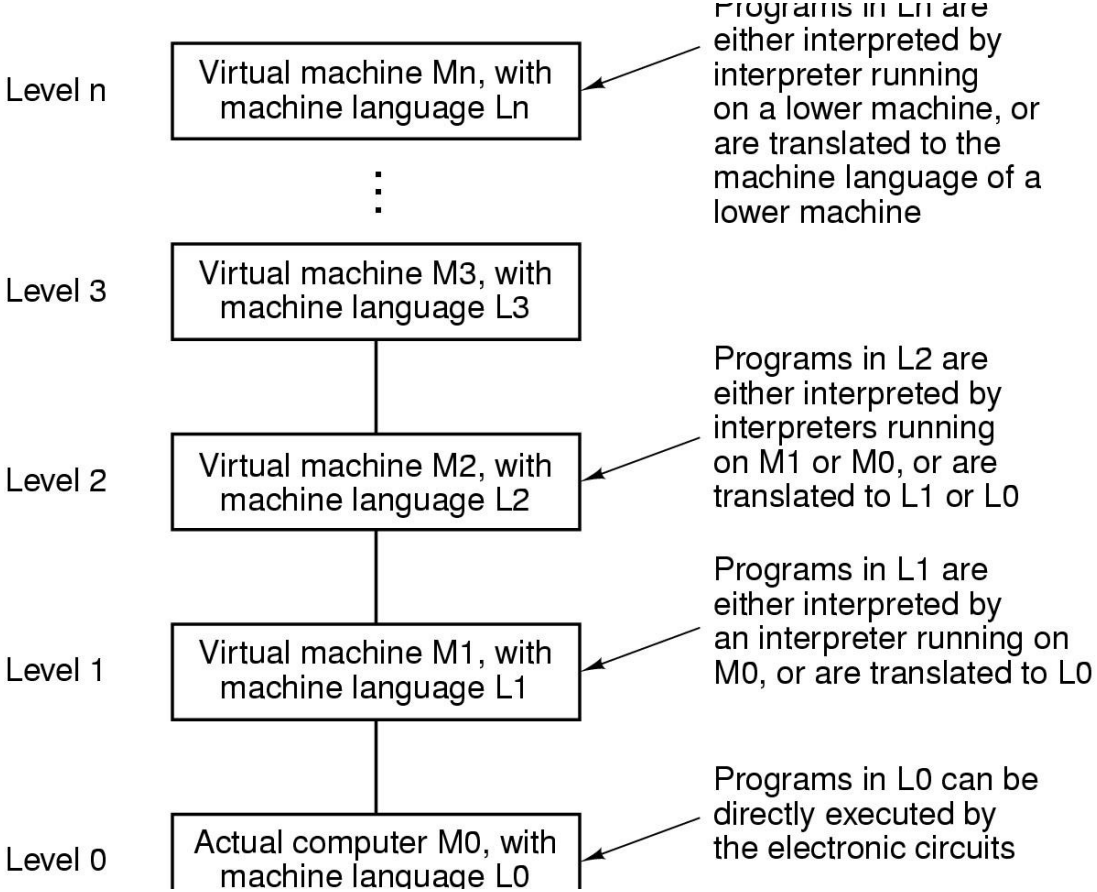
FACULTAD DE INGENIERIA
en acción continua...

Unidad 1 – Arquitectura de computadoras

- Una computadora es capaz de ejecutar programas
- Las instrucciones que el hardware ejecuta directamente son limitadas. Ej:
  - Sumar dos números
  - Verificar si el resultado es 0
  - Copiar datos dentro de la memoria
- Para facilitar el desarrollo de aplicaciones complejas, los **sistemas de cómputo** se estructuran en **capas**
- Cada capa aporta funcionalidad a las capas superiores y se apoya en las capas inferiores

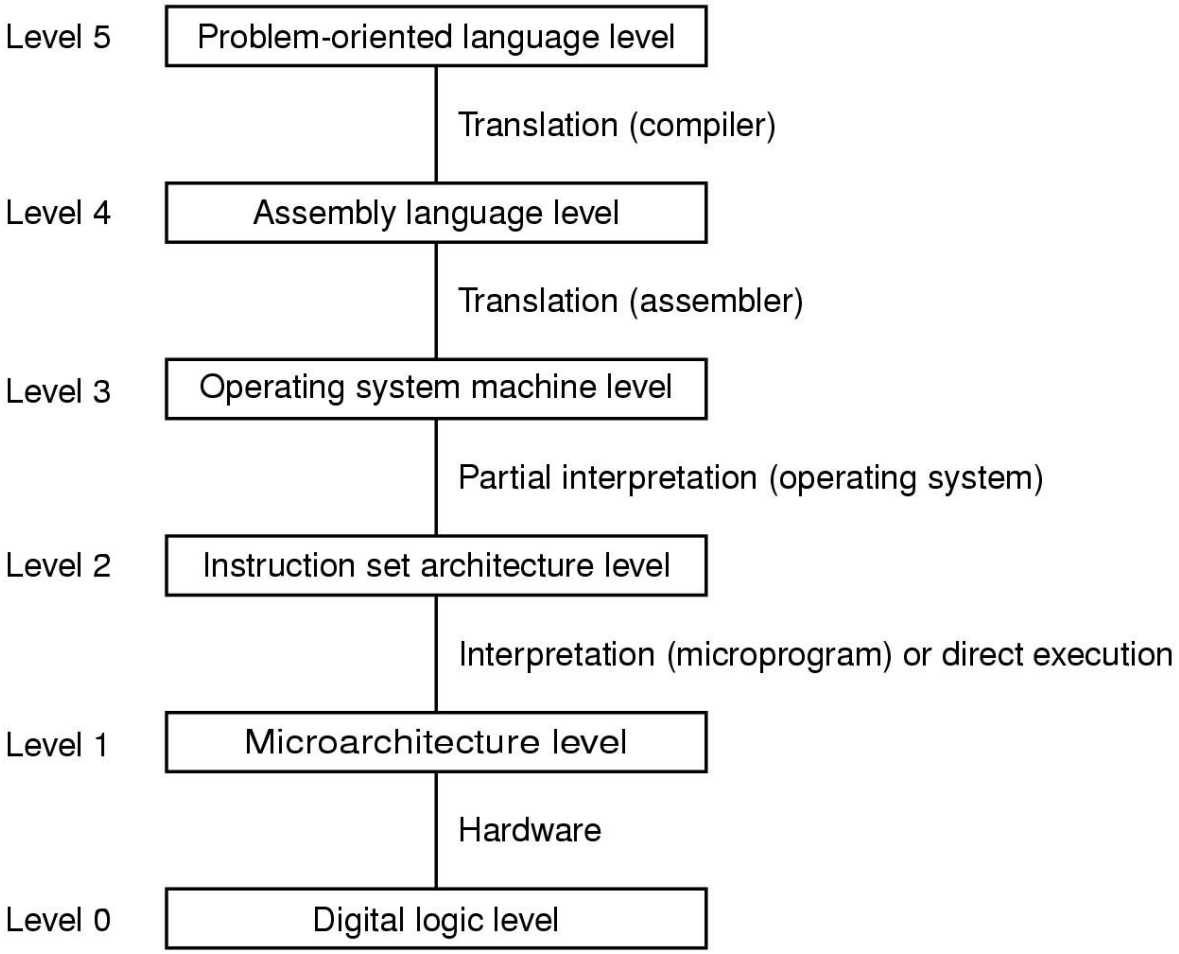
Unidad 1 – Arquitectura de computadoras

- Estructura conceptual de la organización en capas



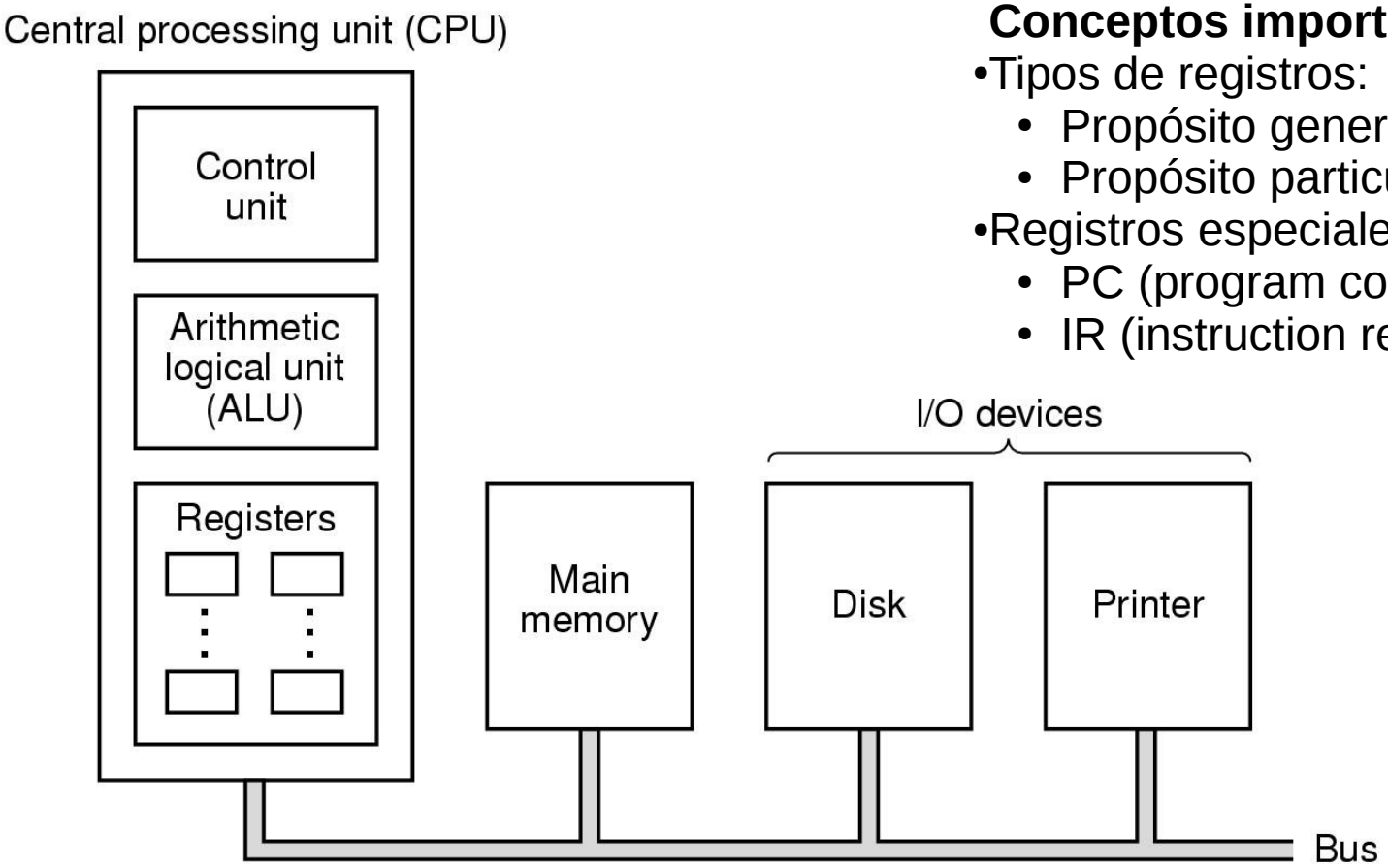
Unidad 1 – Arquitectura de computadoras

- Organización de una máquina multinivel típica



Unidad 1 – Arquitectura de computadoras

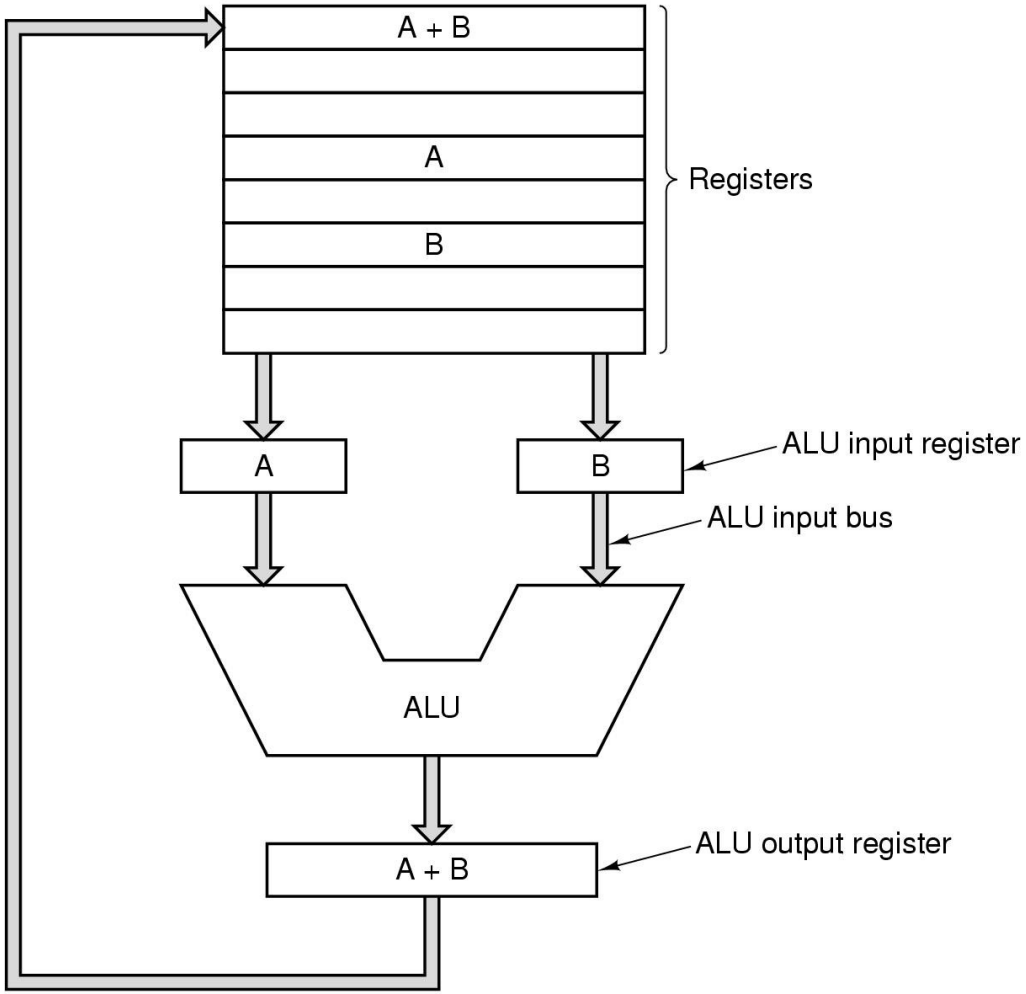
- Componentes principales de una máquina sencilla



Conceptos importantes

- Tipos de registros:
  - Propósito general
  - Propósito particular
- Registros especiales:
  - PC (program counter)
  - IR (instruction register)

- CPU von Neumann típica: camino de datos



Conceptos importantes

- Tipos de Instrucciones
  - Registro-memoria
  - Registro-registro
- Ciclo del camino de datos

Unidad 1 – Arquitectura de computadoras

- Ejecución de instruccciones

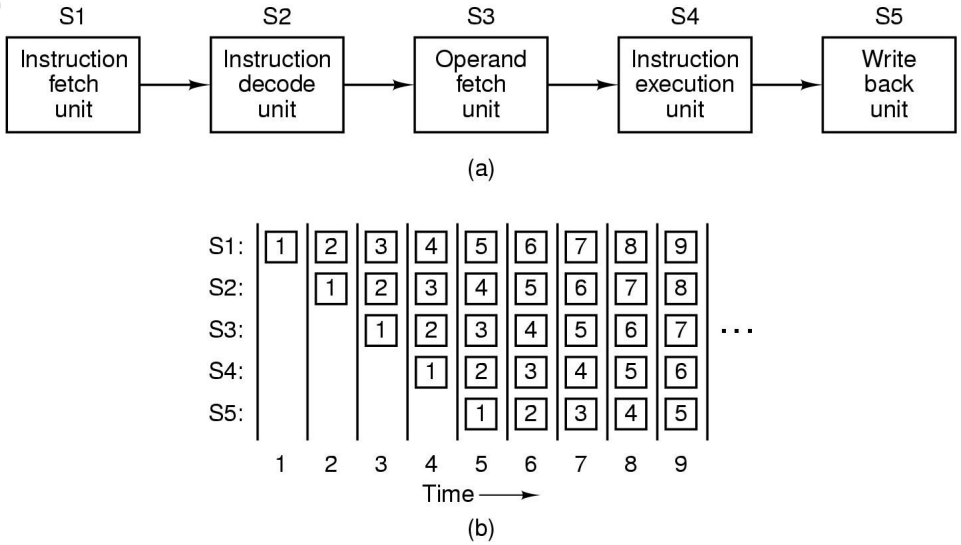
  Cada instrucción se ejecuta mediante una serie de pasos pequeños

  - Buscar la siguiente instrucción en la memoria
  - Actualizar el PC
  - Determinar el tipo de instrucción que se trajo
  - Si la instrucción utiliza una palabra de la memoria, ubicarla
  - Traer la palabra de la memoria a un registro de la CPU (si la instrucción usa una palabra de memoria)
  - Ejecutar la instrucción
  - Volver al primer paso

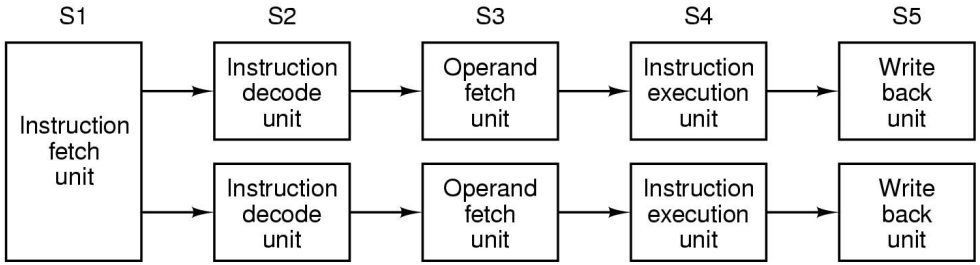
Unidad 1 – Arquitectura de computadoras

- Principios de diseño de CPU
  - RISC vs. CISC
  - Tendencias actuales
    - Ejecutar instrucciones en hardware, sin interpretación por parte del microprograma
    - Maximizar el ritmo de **emisión** de instrucciones
    - Decodificación simple de instrucciones
    - Sólo operaciones de carga/almacenamiento hacen referencia a memoria
    - Incluir registros abundantes

- Paralelismo a nivel de instrucciones
  - Filas de procesamiento
    - Solapamiento de pasos dentro del ciclo del camino de datos

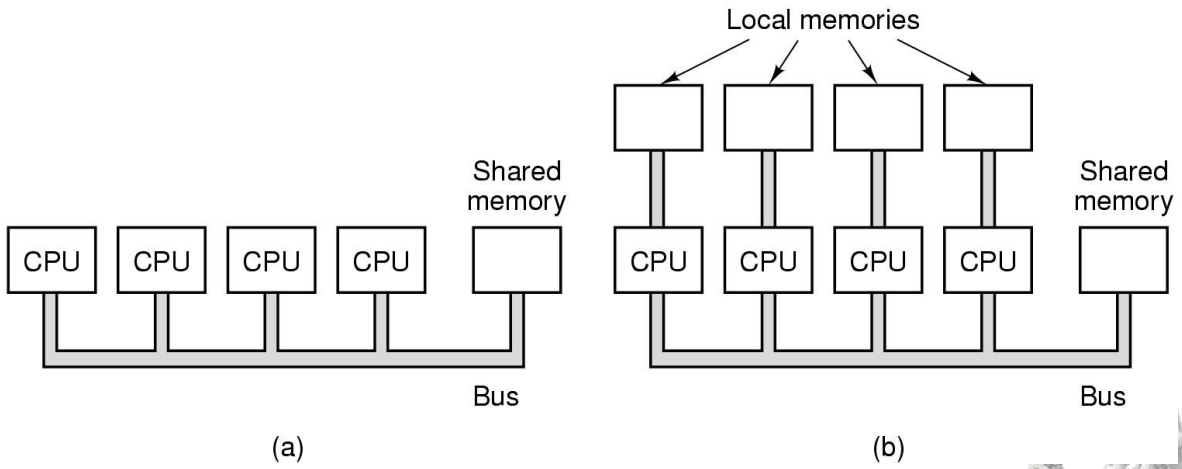


  - Arquitecturas superescalares
    - Múltiples filas de procesamiento (procesadores multi-core)

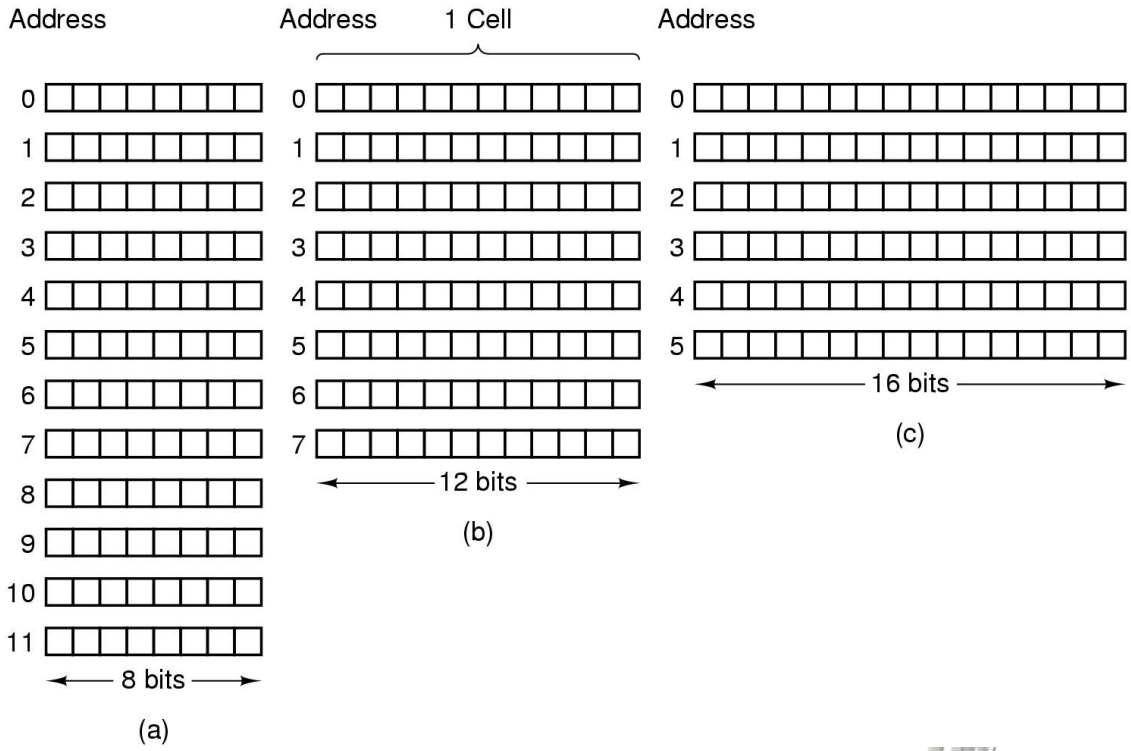


Unidad 1 – Arquitectura de computadoras

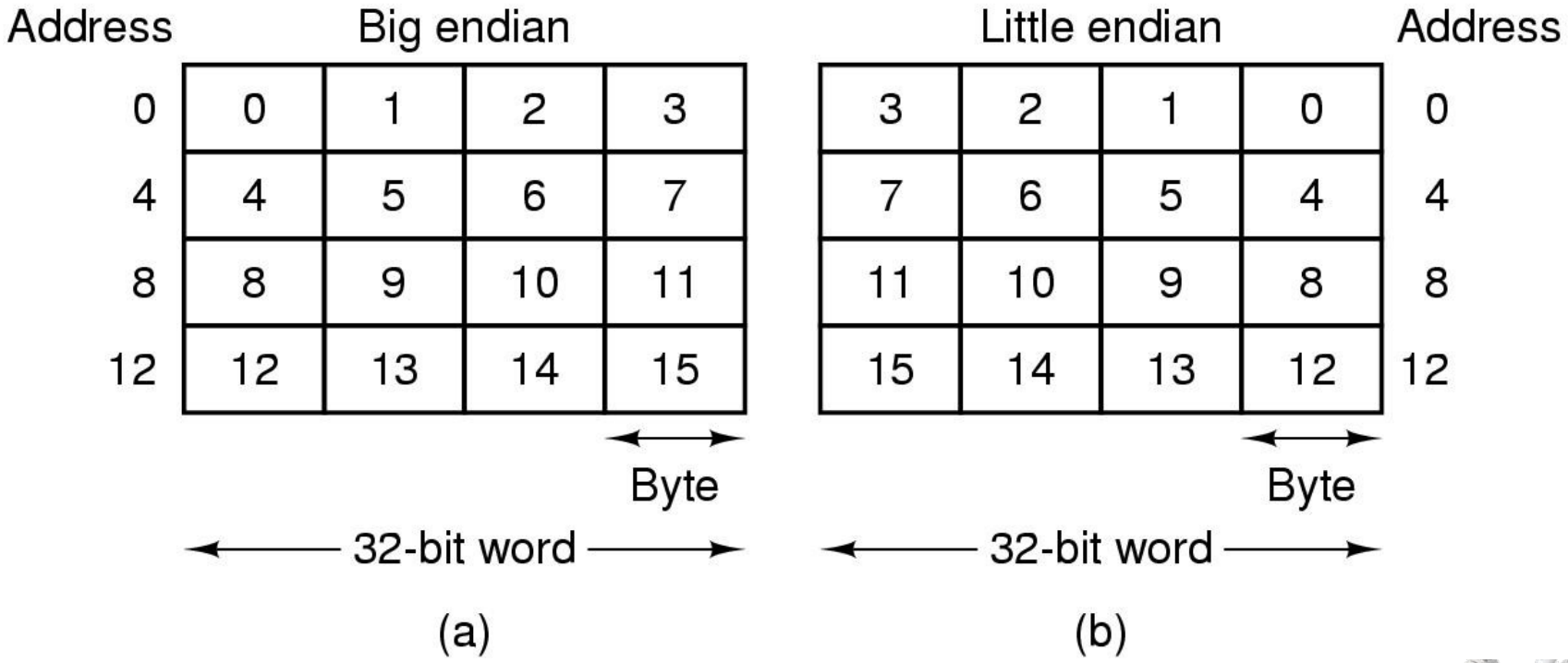
- Paralelismo a nivel de procesador
  - Computadoras de matriz
    - Procesadores que ejecutan el mismo programa con distintos conjuntos de datos (propósito particular)
  - Multiprocesadores
    - Multiples procesadores independientes en un mismo motherboard
  - Multicomputadoras
    - Clustering
    - Grid computing



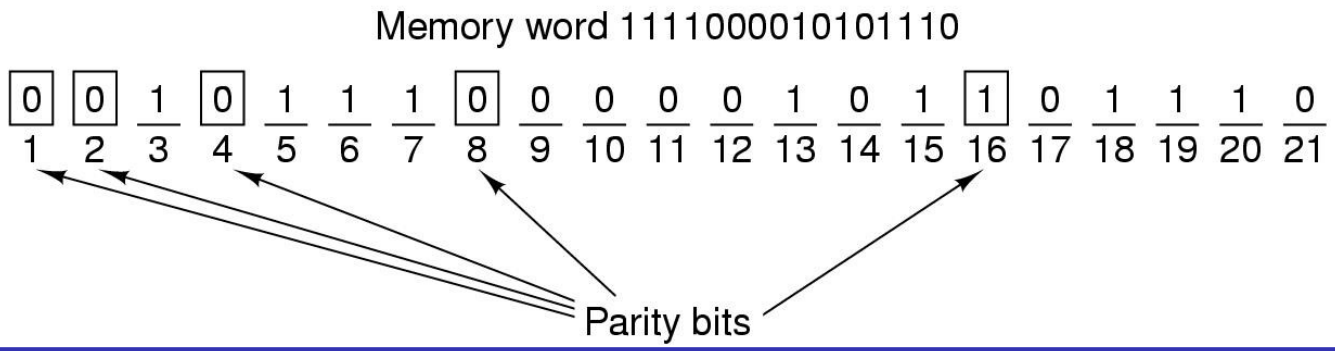
- Memoria
  - Bits, bytes, niveles eléctricos, sistema binario
  - Direcciones de memoria
    - Celdas iguales (llamadas “palabras”)
    - k bits – 2^k combinaciones
    - n direcciones: 0 a n-1
    - Direcciones de m bits: 2^m celdas
    - Celdas son unidad mínima direccionable



- Memoria
  - Ordenamiento de bytes: Little endian vs. Big endian

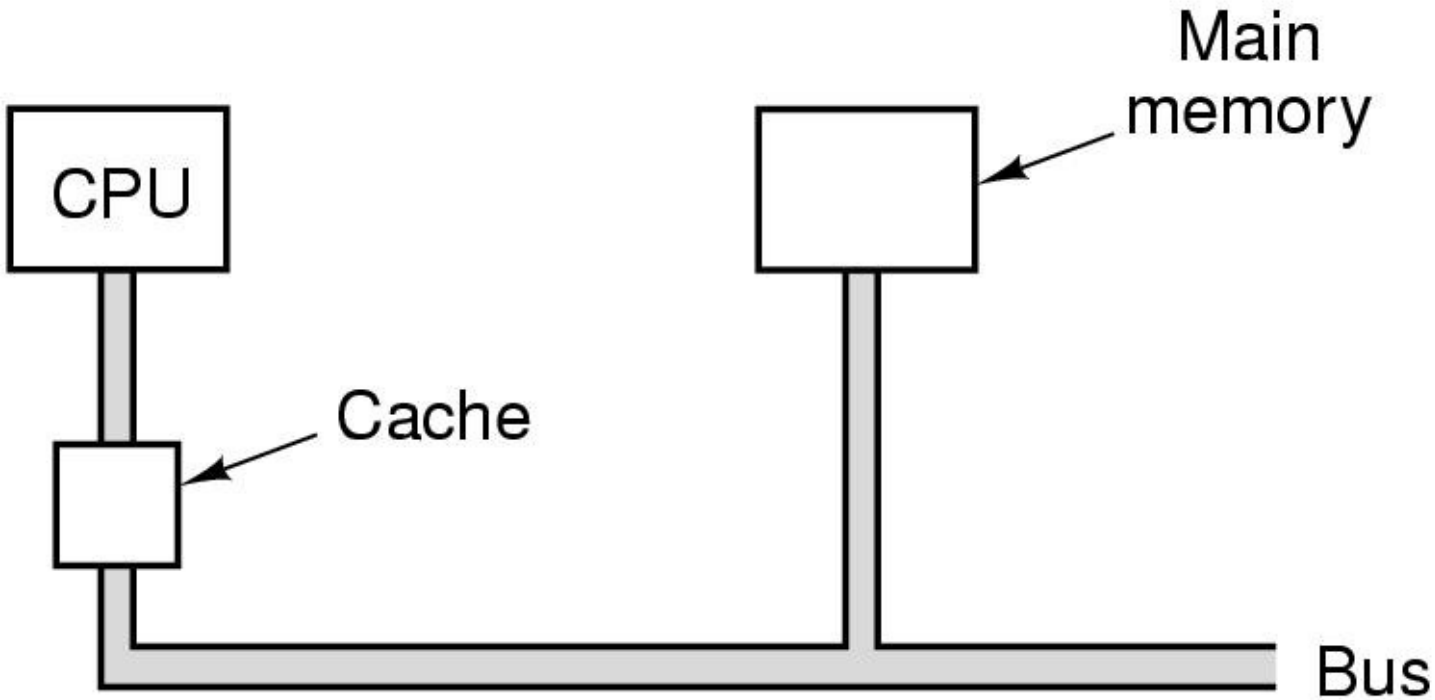


- Memoria: códigos de detección/corrección de errores
  - Agregado de bits de verificación
  - Distancia de hamming: cantidad de bits que separan una palabra válida de otra
  - Ej de código de detección de errores: bit de paridad
    - Sirve para detectar errores de 1 bit
  - Ej. de código de corrección de errores: código de hamming
    - Se insertan varios bits de paridad, en posiciones potencia de 2
    - Cada bit de paridad verifica un subconjunto de la palabra, de manera **redundante**
    - El error se puede corregir verificando cuales bits de verificación corresponden a paridades inválidas



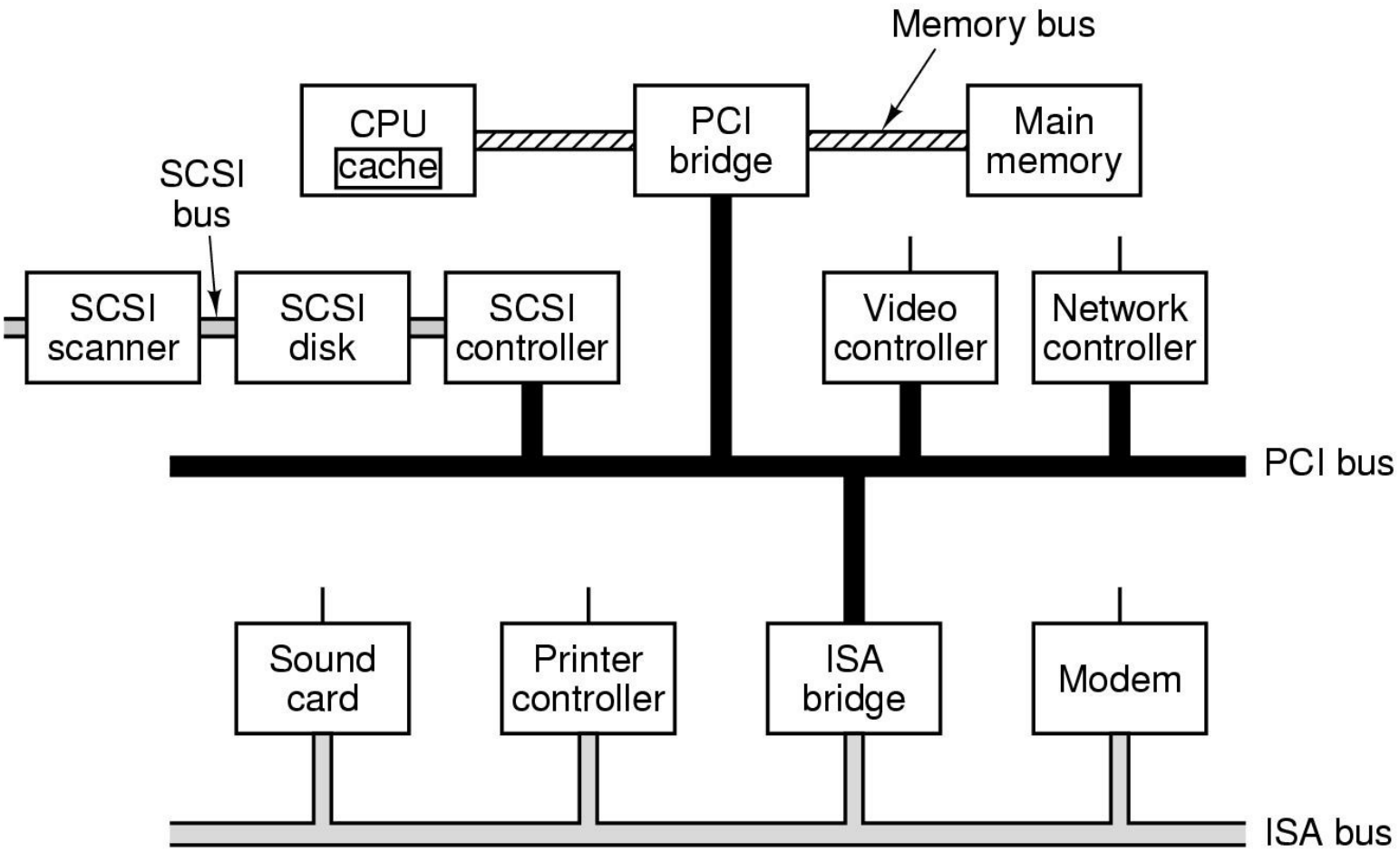
Unidad 1 – Arquitectura de computadoras

- Memoria:
  - Memoria cache



Unidad 1 – Arquitectura de computadoras

- Buses



- NIVEL DE MICROARQUITECTURA

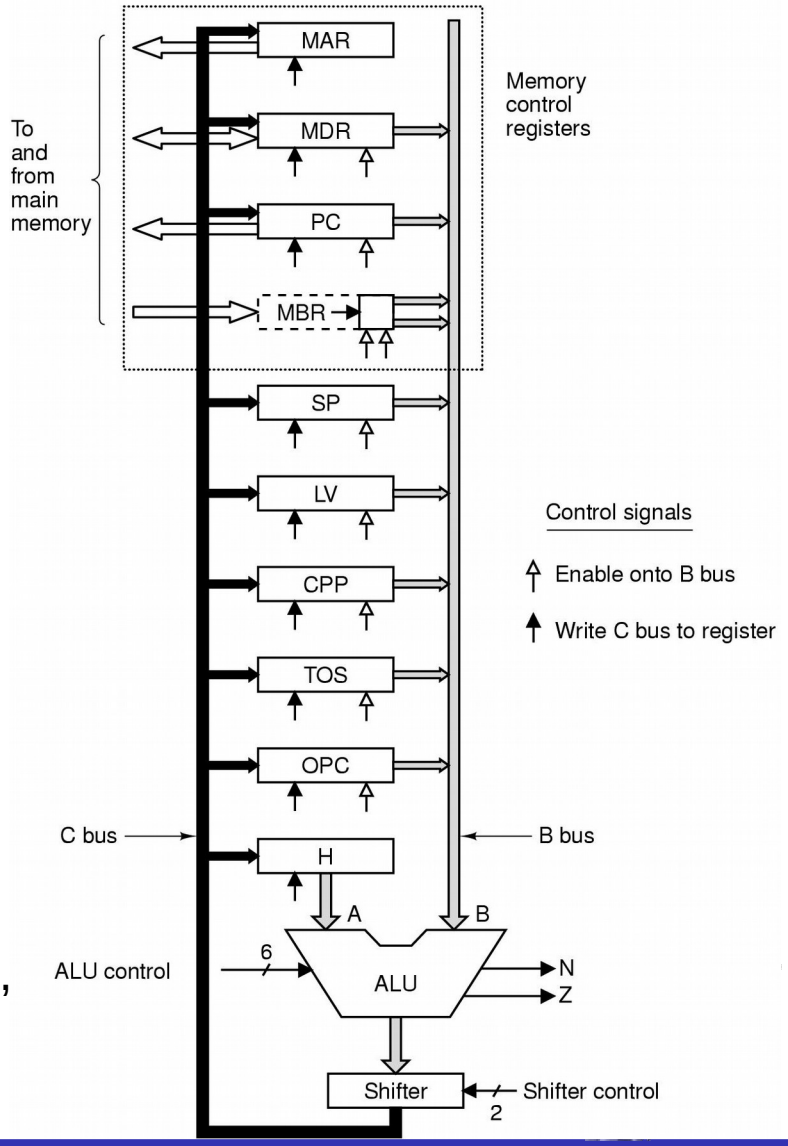
Conceptos importantes

•Un puñado de registros mantienen el estado

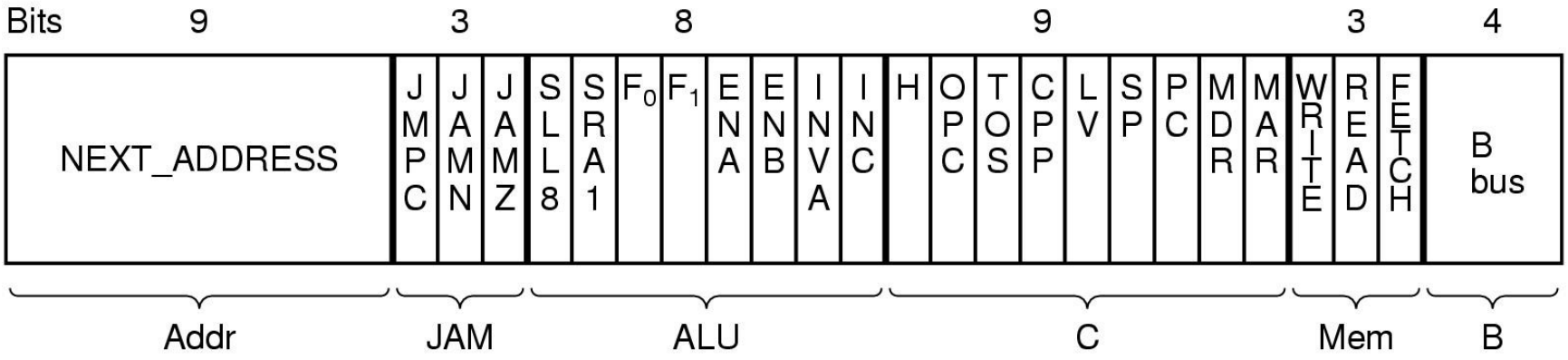
•Cada instrucción tiene
  - un código de operación
  - Cero o más operandos

•El microprograma (o el hardware) ejecuta ciclos de búsqueda-decodificación-ejecución

•El microprograma controla los ciclos de la trayectoria de datos mediante señales (ej: habilitar registros, insertar una dirección de memoria en MAR, activar entradas de la ALU, etc)



- Ejemplo de microinstrucción
  - Incluye el estado de todas las señales de control

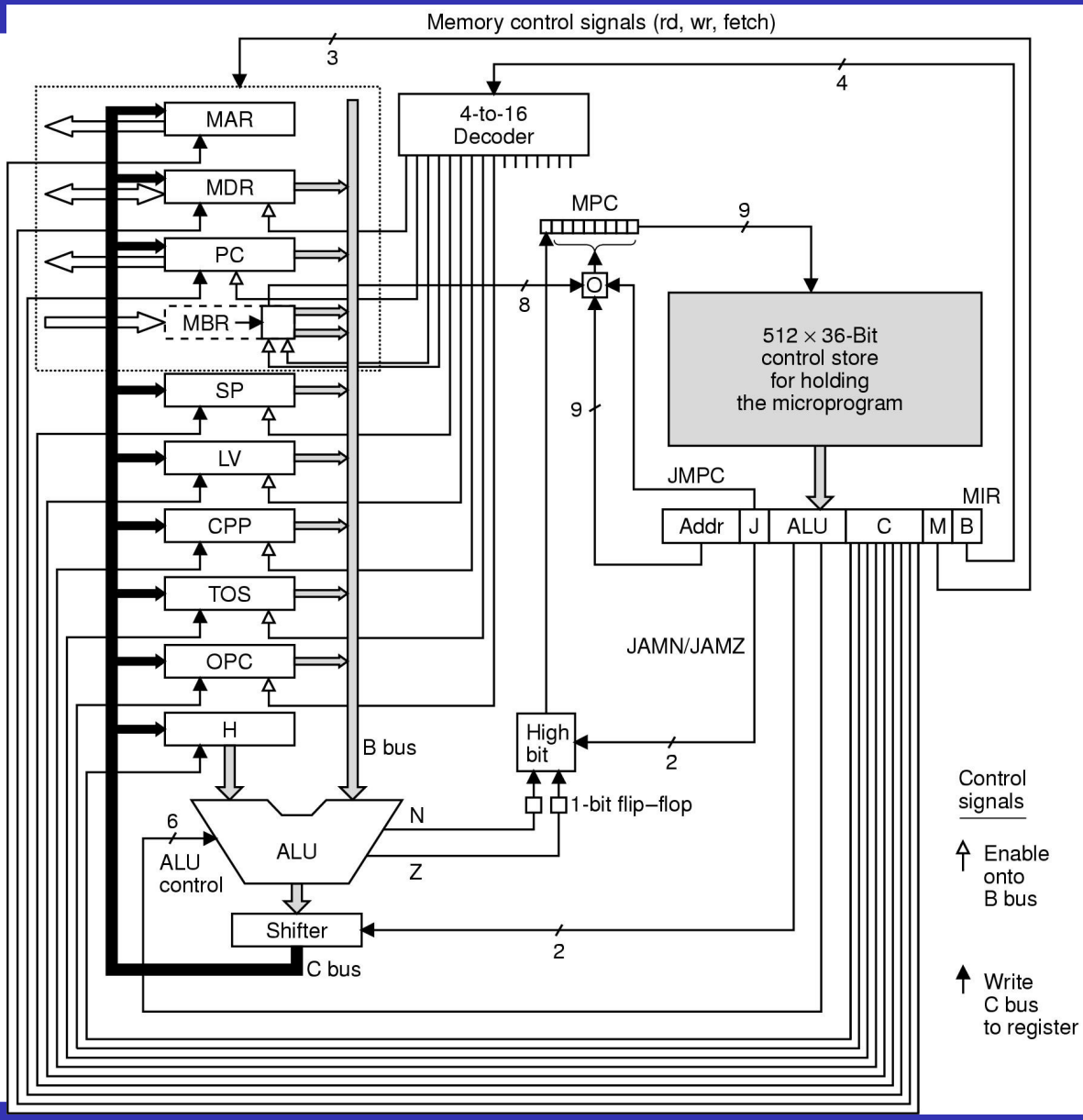


B bus registers

0 = MDR	5 = LV
1 = PC	6 = CPP
2 = MBR	7 = TOS
3 = MBRU	8 = OPC
4 = SP	9 -15 none

Unidad 1 – Arquitectura de computadoras

- Lógica de control de la trayectoria de datos
  - Un ***secuenciador*** recorre la ***secuencia de operaciones*** necesarias para ejecutar la instrucción del nivel ISA



- NIVEL ISA (INSTRUCTION SET ARCHITECTURE)
  - Las instrucciones del nivel ISA son generadas por un compilador o un ensamblador
  - En el nivel ISA solamente se “ven” los detalles que son útiles para el desarrollador de compiladores/ensambladores
  - En el nivel ISA generalmente se proveen al menos 2 modos
    - Modo Kernel → permite ejecutar el SO de manera “protegida”
    - Modo Usuario → permite ejecutar programas de usuario, restringiendo la ejecución de instrucciones “peligrosas” o el acceso a sectores de memoria restringidos (ej: dispositivos de E/S)

- NIVEL ISA (INSTRUCTION SET ARCHITECTURE)
- Registros
  - Los registros pueden ser
    - de propósito particular (ej: el PC no puede utilizarse para otra cosa que para apuntar a la siguiente instrucción a ejecutar)
    - de propósito general: estos permiten realizar cálculos y almacenar resultados temporalmente, dentro de la CPU
  - Existen algunos registros de propósito especial que solo pueden ser accedidos en modo kernel (es decir, solo por el sistema operativo)
    - Estos registros controlan caches, memoria, dispositivos de E/S, etc.

- NIVEL DEL SISTEMA OPERATIVO
- El sistema operativo se puede ver como un programa que añade instrucciones y funciones nuevas, más allá de las que provee el nivel ISA
- El sistema operativo se implementa casi siempre en software
- El sistema operativo proporciona las funciones a las que pueden acceder los programadores de aplicaciones
  - Incluye casi todas las instrucciones y funciones del nivel ISA
  - Agrega funciones adicionales, denominadas **llamadas al sistema**
- Las llamadas al sistema son funciones que permiten operar la máquina subyacente de manera más simple y potente, permitiendo
  - Manejar memoria virtual (paginación y segmentación)
  - Ejecutar múltiples programas simultáneamente
  - Acceder a los dispositivos de E/S más fácilmente

- NIVEL DEL LENGUAJE ENSAMBLADOR
  - Los programas de aplicación se escriben mediante lenguajes simbólicos (lenguaje fuente)
  - Los programas simbólicos se **traducen** al lenguaje objetivo, que generalmente es un lenguaje numérico
  - Existen básicamente 2 tipos de traductores
    - Ensambladores
    - Compiladores
  - Una alternativa a la traducción es la **interpretación**
  - El lenguaje ensamblador
    - Está “más cerca” del lenguaje en nivel ISA que los lenguajes de alto nivel
    - Es más dependiente de la máquina subyacente que los lenguajes de alto nivel (es más difícil portarlo a otras plataformas)

- NIVEL DEL LENGUAJE ENSAMBLADOR
  - Razones para utilizar lenguaje ensamblador
    - Rendimiento (tiempo de procesamiento, fine tune de ciertas características de la arquitectura ISA, etc.)
    - Acceso al hardware de manera más amplia
    - Espacio para instrucciones muy limitado (ej: en microcontroladores)
  - Algunos compiladores de algunos lenguajes (ej: Compilador GCC para C) permiten mezclar lenguaje ensamblador con el lenguaje de alto nivel: a esto se lo llama “embeber un lenguaje dentro de otro”.

- NIVEL DEL LENGUAJE ENSAMBLADOR
  - Cada sentencia en lenguaje ensamblador tiene 3 partes:
    - Etiqueta
    - Código de operación
    - Operandos (cero o más)
  - Algunos ejemplos:

```
FORMULA:  MOV     EAX, I
          ADD     EAX, J
          MOV     N, EAX
          JMP     FORMULA
I         DW      3          ; Reserva 4 bytes inicializados a 3
J         DW      4
N         DW      0
```

Unidad 1 – Arquitectura de computadoras

- Sistemas de numeración, sistema binario y precisión
  - El principal problema al hacer cálculos con computadoras es la precisión
  - Una computadora tiene una cantidad **fija** y **finita** de dígitos para almacenar valores y operar con ellos
  - La cantidad de bits para trabajar se define al momento de diseñar la máquina
- Problema de la precisión finita: cerradura. Violaciones
  - El resultado de un cálculo puede ser más grande o más chico que el maximo/minimo posible
  - El resultado puede simplemente no estar en el conjunto de lo representable (ej: números decimales)
- Estos problemas causan que algunas propiedades aritméticas no se cumplan siempre (ej: asociatividad)

- Sistemas de numeración, sistema binario y precisión
- Ejemplos:
  - Precisión de 3 dígitos decimales
  - Representa números del 0 al 999
  - 500+500 = 1000 ; fuera del rango
  - 500-501 = -1; fuera del rango
  - 500 + (600-200) = (500 + 600) – 200 → aritmética tradicional

    500 + 400 = <desbordamiento> - 200 → no se cumple en la aritmética de precisión finita

- Sistemas de numeración, sistema binario y precisión
- Sistemas de numeración con base
  - Se apoyan sobre la posición de los dígitos
  - Base decimal:

    $N = d_n \ldots d_2\ d_1\ d_0\ .\ d_{-1}\ d_{-2}\ d_{-3} \ldots d_{-k}$

    $$N = \sum_{i=-k}^{n} d_i 10^i$$

  - De manera similar para la base binaria:

    $N = d_n \ldots d_2\ d_1\ d_0\ .\ d_{-1}\ d_{-2}\ d_{-3} \ldots d_{-k}$

    $$N = \sum_{i=-k}^{n} d_i 2^i$$

Unidad 1 – Arquitectura de computadoras

- Sistemas de numeración, sistema binario y precisión
- Las bases más importantes en informática son la binaria, octal y hexadecimal
- Un sistema de numeración de base ***k*** requiere k símbolos para representar sus números
  - Decimal: números del 0 al 9
  - Binario: 0 y 1
  - Octal: números del 0 al 7
  - Hexadecimal: números del 0 al 9 y letras de la A a la F
- Ejemplos
- Conversión entre bases

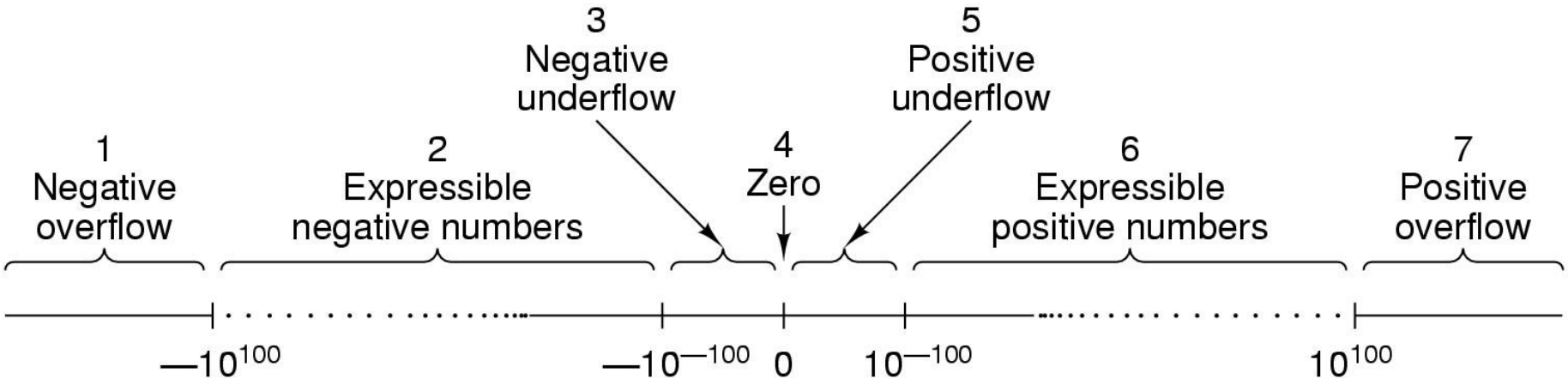
Unidad 1 – Arquitectura de computadoras

- Números binarios negativos
  - Magnitud con signo
  - Complemento a 1: Invertir 0s por 1s
  - Complemento a 2
    - Invertir 0s por 1s y sumar 1
    - Si hay desbordamiento, el bit de la izquierda se descarta
  - Exceso de 2^{m-1} (para sistemas de m bits)
    - Se “desplaza” el 0 a la mitad del espacio de números
    - Ej: m=8 bits, el rango es de 0 a 255, y representa nros de -128 a 127
    - Ej: para m=8 bits (exceso 2^7=128), -5 → -5 + 128 = 123
      123 → 01111011 (+5 → 00000101)
  - Propiedades deseables:
    - 1 sólo símbolo para el 0
    - Simetría (igual cantidad de nros negativos/positivos)

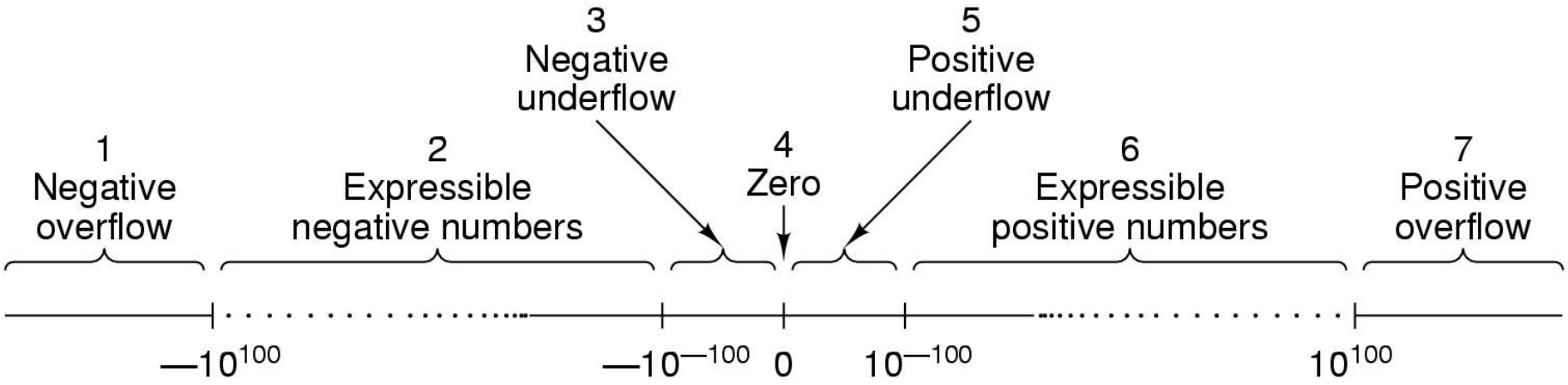
Unidad 1 – Arquitectura de computadoras

- Números de punto flotante
  - Problemas relevantes:
    - Intervalo (número máximo/mínimo que puede representarse)
    - Precisión (cantidad de cifras significativas)
  - Estos problemas derivan de la naturaleza **finita** de las computadoras
  - Características deseables
    - Amplio rango
    - Precisión lo más grande posible, sin desperdicio de espacio
  - Esquema adoptado: números con punto flotante
    - Similar a la notación científica
    - Tienen 3 elementos:
      - ***Signo*** del número
      - ***Exponente*** (signo + magnitud)
      - Magnitud de la fracción (***mantisa***)

- Números de punto flotante
  - Inconvenientes que se presentan en la aritmética de punto flotante
    - Desbordamiento negativo (negative overflow)
    - Subdesbordamiento negativo (negative underflow)
    - Subdesbordamiento positivo (underflow)
    - Desbordamiento positivo (overflow)
  - Los subdesbordamientos son menos graves que los desbordamientos, porque el 0 es una buena aproximación para resolver el problema



- Números de punto flotante: Diferencias con los números reales
  - Los nros de PF no pueden representar valores en las regiones 1, 3, 5 y 7
  - Entre 2 números reales hay infinitos números reales; entre 2 números de PF **adyacentes**, no hay ningún número de PF
    - Si un cálculo cae “entre” 2 números válidos, se adopta el número válido más cercano al resultado → ***redondeo***
  - La distancia entre 2 números adyacentes de PF no es constante



Unidad 1 – Arquitectura de computadoras

- Números de punto flotante: Estándar IEEE 754

Normalized: $0\ \underbrace{1\,0\,0\,1\,0\,0\,1}\ .\ \underbrace{1\ 1\ 0\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0}\ = 2^9\,(1\times2^{-1}+1\times2^{-2}+1\times2^{-4}+1\times2^{-5}) = 432$

Sign + ; Excess 64 exponent is $73-64=9$

Fraction is $1\times2^{-1}+1\times2^{-2}+1\times2^{-4}+1\times2^{-5}$