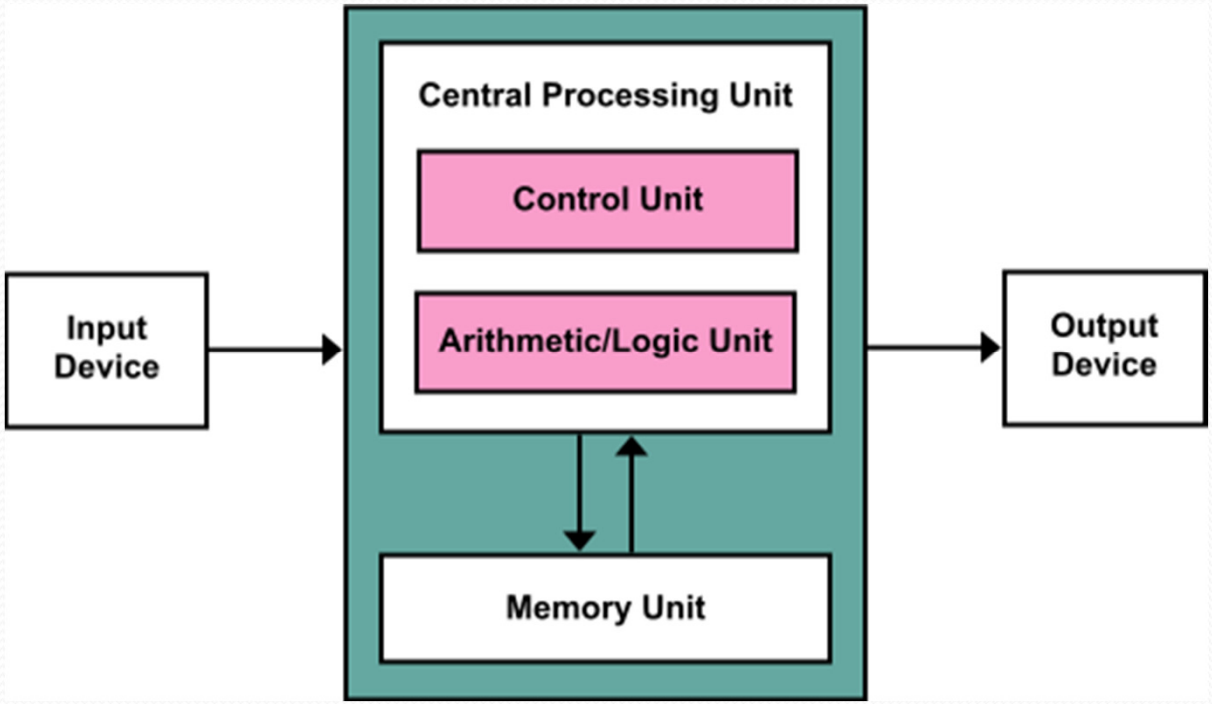
Organización de computadoras

Conceptos básicos

Modelo de von Neumann

- También conocida como arquitectura de Princeton.
- Propuesta por John von Neumann en 1945.
- Partes de una computadora digital:
  - Unidad de procesamiento (CPU – unidad central de procesamiento).
  - Memoria.
  - Almacenamiento externo.
  - Mecanismos de entrada y salida (I/O).

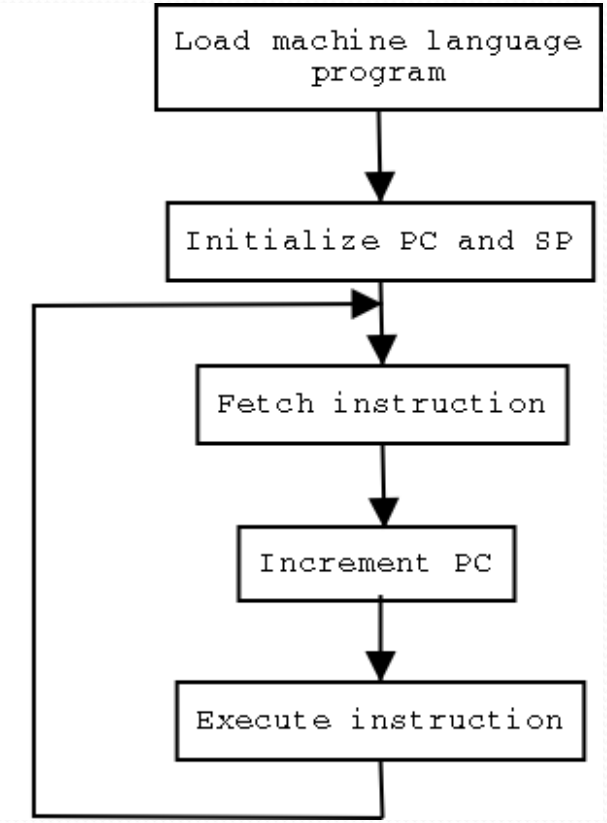
Modelo de von Neumann



- **Fuente: https://en.wikipedia.org/wiki/Von_Neumann_architecture#/media/File:Von_Neumann_Architecture.svg (CC BY-SA 3.0)**

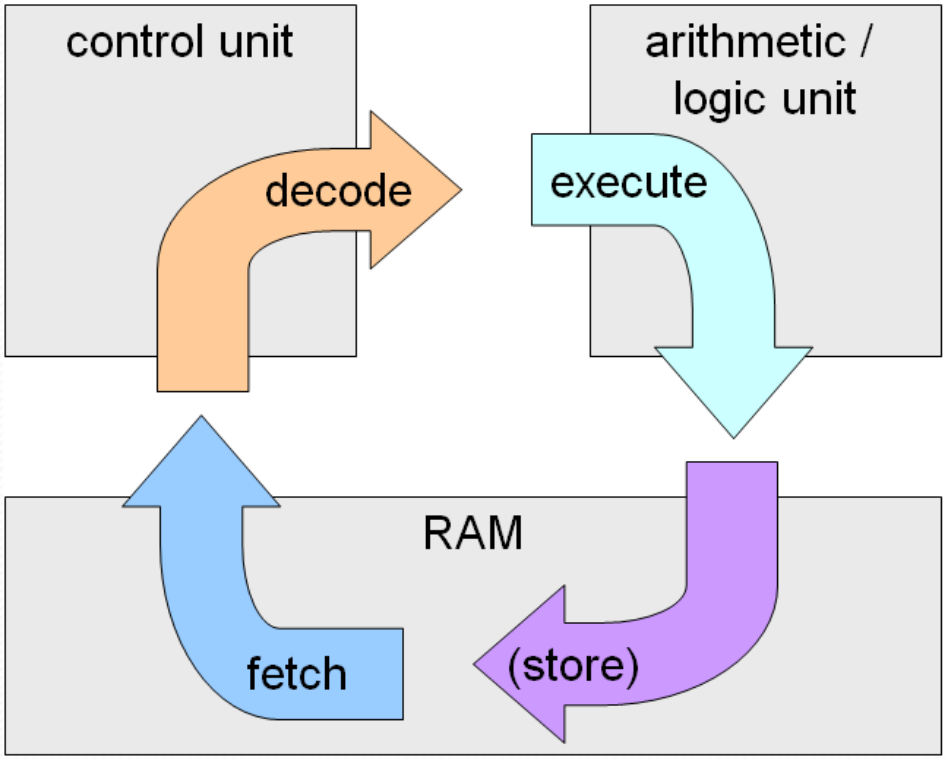
Modelo de von Neumann

- Ciclo de instrucción:



- **Fuente:** http://faculty.cooper.edu/smyth/cs225/images/von_neumann_highlevel.png

Modelo de von Neumann



- **Fuente: https://samitboony.files.wordpress.com/2013/11/fetch-execute-cycle.png**

Cuello de botella de von Neumann

- Solo hay un bus entre la CPU y la memoria.
- En un ciclo dado solo se puede transferir una instrucción o un dato.
- Limita la velocidad (rendimiento).
- Solución: usar dos memorias, una de datos y otra de instrucciones (arquitectura de Harvard).
- Las CPUs modernas usan una arquitectura *modificada* de Harvard.

Cuello de botella de von Neumann

- Los datos e instrucciones están en la misma memoria principal (arquitectura von Neumann o Princeton).
- Los datos e instrucciones están separados en el caché (arquitectura Harvard).

Conjunto de instrucciones

- También conocido como ISA (instruction set architecture).
- Conjunto de instrucciones que ofrece una familia de procesadores (lenguaje ensamblador).
- Familias:
  - x86 – Intel.
  - ARM – ARM Holdings.
  - MIPS – Wave Computing.
  - Motorola 68000 – Motorola.
  - DEC Alpha – Digital Equipment Corporation.

Conjunto de instrucciones

- Cada instrucción es directamente ejecutada por el hardware.
- Se representa con un formato binario. El hardware solo entiende bits.
- Los objetos físicos son bits, bytes, palabras (words).
- Tamaño típico de palabra: 4 u 8 bytes (32 o 64 bits).

Conjunto de instrucciones

- Dos opciones de tamaño de las instrucciones:

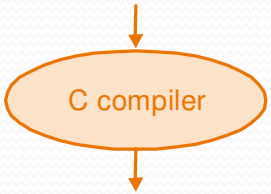
- Fijo. Cada instrucción ocupa el mismo número de bytes.
- Variable. Cada instrucción ocupa distintos números de bytes.

Abstracción

Bajando el nivel de abstracción revela otras información

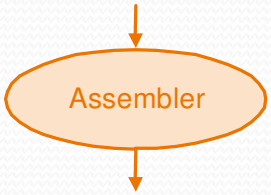
High-level language program (in C)

```
swap(int v[], int k)
{int temp;
  temp = v[k];
  v[k] = v[k+1];
  v[k+1] = temp;
}
```



Assembly language program (for MIPS)

```
swap:
    muli $2, $5,4
    add  $2, $4,$2
    lw   $15, 0($2)
    lw   $16, 4($2)
    sw   $16, 0($2)
    sw   $15, 4($2)
    jr   $31
```



Binary machine language program (for MIPS)

```
00000000101000010000000000011000
00000000100011100001100000100001
10001100011000100000000000000000
10001100111100100000000000000100
10101100111100100000000000000000
10101100011000100000000000000100
00000011111000000000000000001000
```

Ejemplo ampliado

```
swap(int v[], int k)
{ int temp;
    temp = v[k]
    v[k] = v[k+1];
    v[k+1] = temp;
}
```

```
swap:
    muli $2, $5, 4
    add $2, $4, $2
    lw $15, 0($2)
    lw $16, 4($2)
    sw $16, 0($2)
    sw $15, 4($2)
    jr $31
```

Instrucciones

- Por lo general son operaciones simples que pueden ser ejecutadas directamente por el hardware.
- Se identifican por un opcode (código de operación).
- Ejemplo de MIPS:

```
add $s0, $s1, $s2
```

- El opcode es “add” (suma)
- Tienen operandos: 0, 1, 2 o 3.
- La instrucción de arriba tiene 3 operandos.
- Significa s0 = s1 + s2

Instrucciones

- Dos opciones de indicar operandos:

- Implícito. El opcode implica la dirección de los operandos.
- Ejemplo en MIPS:

  syscall
- La instrucción hace una llamada al sistema.
- El número de servicio debe estar en el registro v0.

Instrucciones

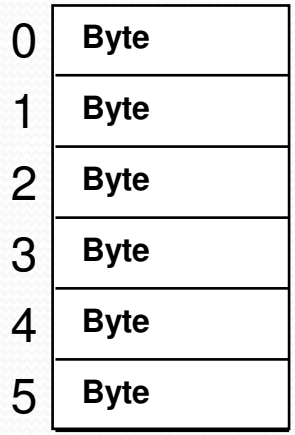
- Explícito. Las direcciones vienen en los operandos.
- Ejemplo de MIPS:

  add $s0, $s1, $s2

- Dos operandos fuentes: s1 y s2
- Un operando destino: s0
- s0 = s1 + s2

Organización de la memoria

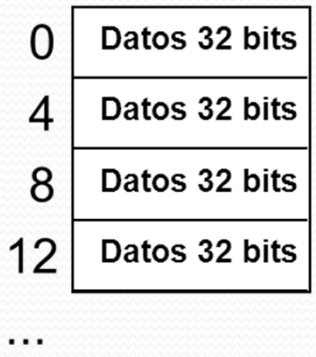
- Se puede ver como un vector (arreglo de una dimensión) de un byte.
- Una dirección de memoria es un índice del vector.



Con *n* bits se pueden direccionar 2^n bytes.
Rango de direcciones: 0 – 2^n - 1

Organización de la memoria

- Alineación. La dirección de una palabra en memoria comienza en un múltiplo del número de bytes que ocupa una palabra.
- Ejemplo de memoria con palabras alineadas de 4 bytes.



Byte-addressing

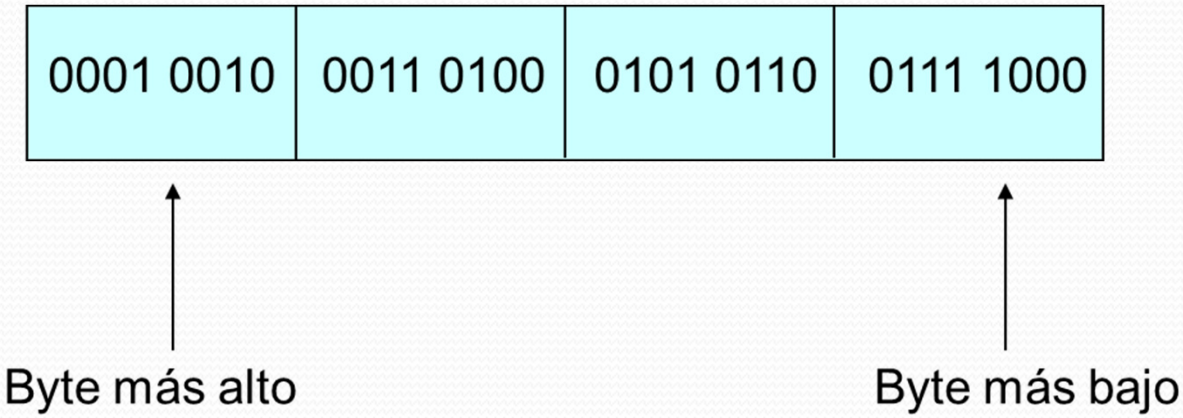
- La memoria es un arreglo de bytes.
- Las direcciones de memoria hacen referencia a un byte.
- La dirección de una palabra (conjunto de bytes) es la dirección de su primer byte.
- La mayoría de las CPUs modernas utilizan byte-addressing.

Big-endian/Little-endian

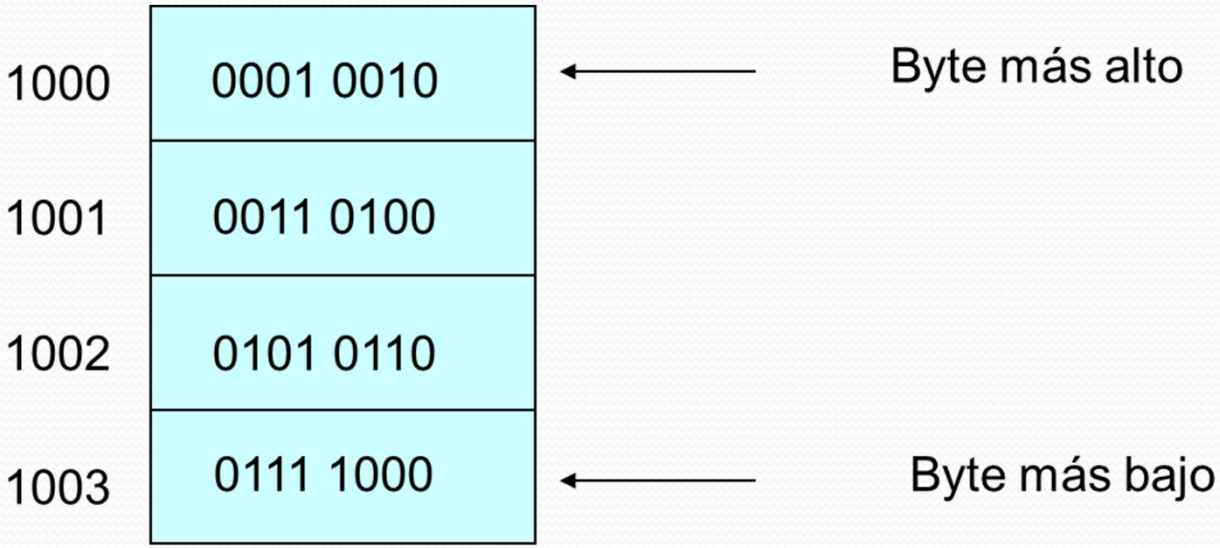
- Hay dos formas de guardar una palabra multi-byte en memoria:
  - Big-endian. El byte más alto (más significativo) se guarda en la dirección mas baja.
  - Little-endian. El byte más bajo (menos significativo) se guarda en la dirección mas baja.

Big-endian/Little-endian

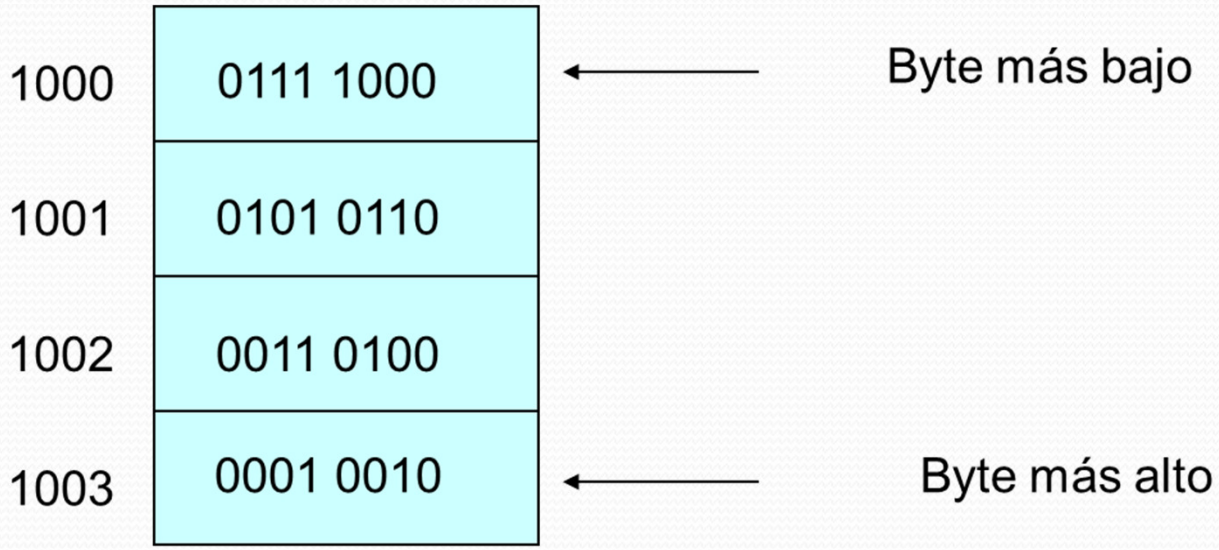
- Ejemplo: guardar la palabra de 4 bytes 12345678_{16} en la dirección 1000.



Big-endian


1000
0001 0010
1001
0011 0100
1002
0101 0110
1003
0111 1000
Byte más alto
Byte más bajo

Little-endian



Big-endian/Little-endian

- Intel x86 es little-endian.
- Sun SPARC es big-endian.

RISC/CISC

- RISC (Reduced Instruction Set Architecture) es una estrategia de diseño de CPUs donde cada instrucción tiene una sola función y se ejecuta de manera rápida.
- Es lo contrario de CISC (Complex Instruction Set Architecture), donde cada instrucción hace muchas funciones.

RISC/CISC

- Ejemplos de RISC incluyen Alpha, ARC, ARM, AVR, MIPS, PA-RISC, PIC, PowerPC, SuperH, and SPARC.
- Ejemplos de CISC incluyen las familias Motorola 68000 e Intel 80x86.

Instrucciones típicas

- Aritméticas: suma (add), resta (sub), multiplicación (mul), división (div).
- Lógicas: and, or, xor, not.
- Movimiento de datos: copia (copy), carga (load), guarda (store), mueve (move).
- Control: brinca (jump), brinca condicional (jump if), llama subrutina (call), regresa de subrutina (ret).
- Sistema: llamadas a funciones del sistema operativo.