

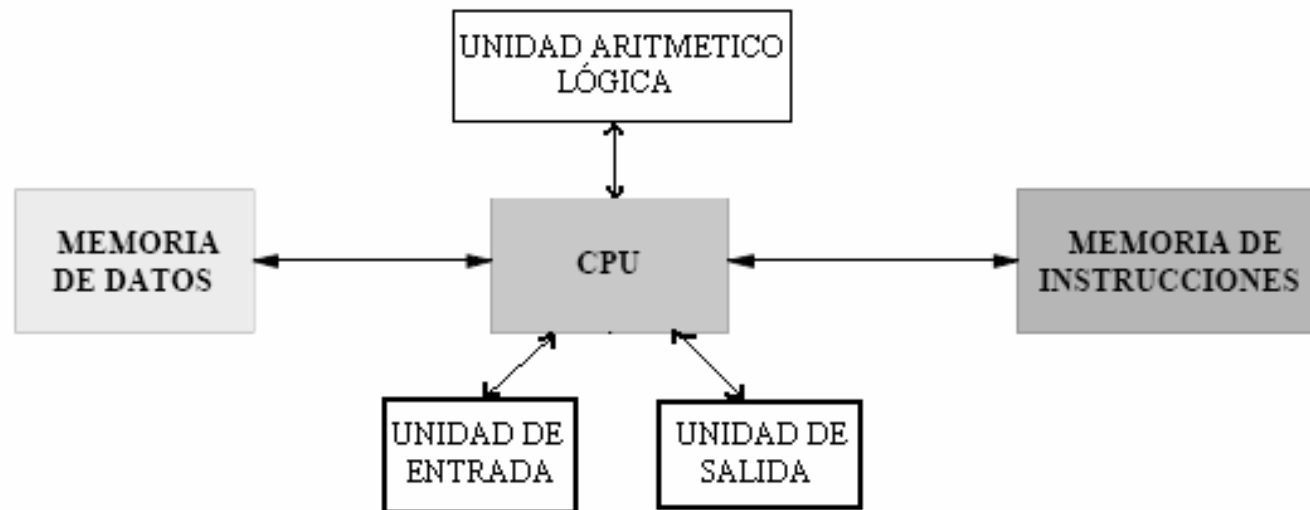
66.70 Estructura del Computador

Arquitectura del Set de Instrucciones

Instruction Set Architecture (ISA)

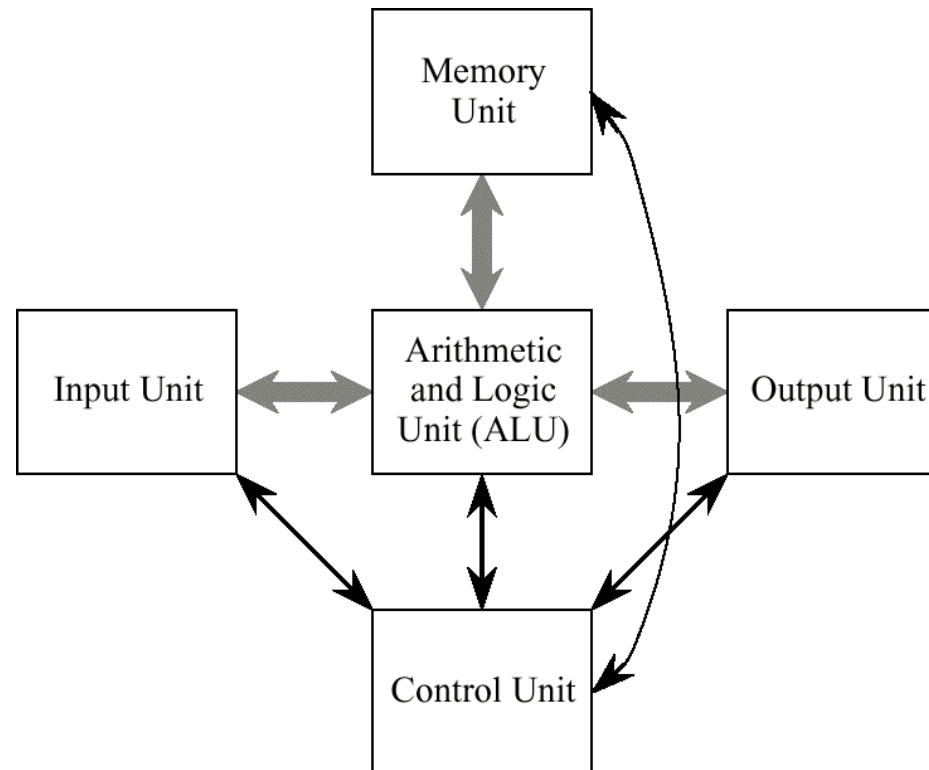
Arquitectura Harvard

- ❖ Responde a la arquitectura de las primeras computadoras
- ❖ Se aplica actualmente en sistemas pequeños (p.e.: electrodomésticos)



Arquitectura von Neumann

- ✓ Unifica el almacenamiento físico de datos y programa



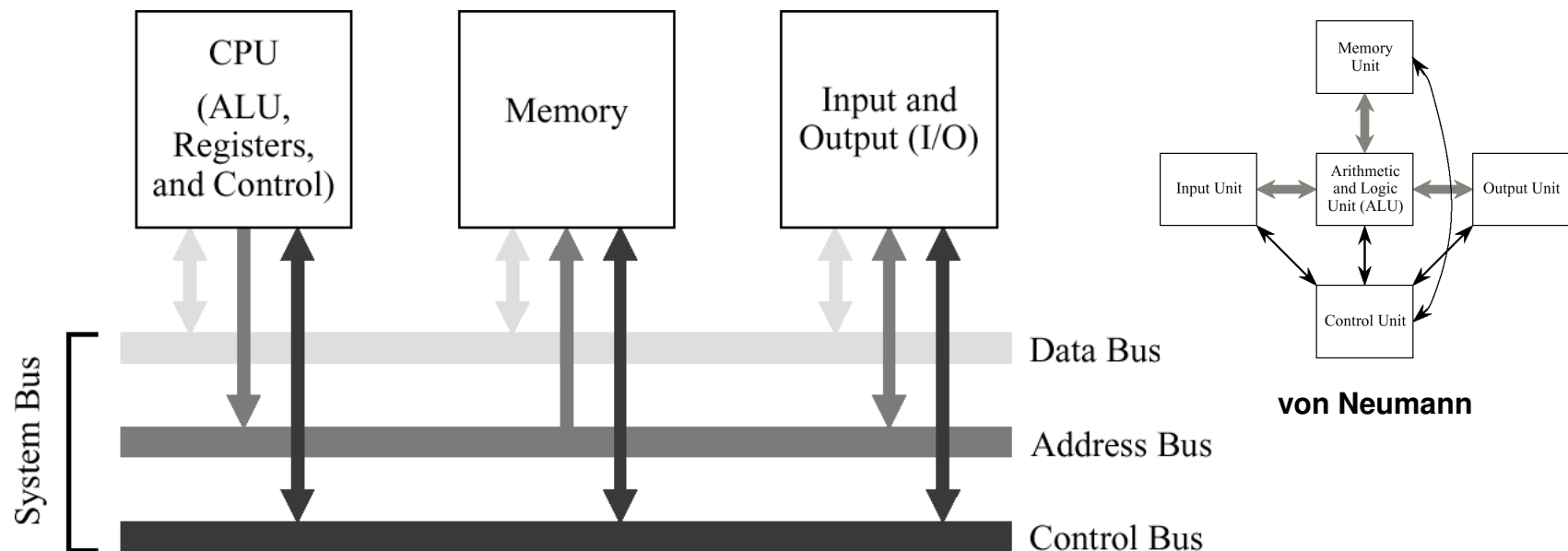
Arquitecturas von Neumann vs. Harvard

- ❖ Con la “arquitectura von Neumann” se logra modificar programas con la misma facilidad que modificar datos, permitiendo:
 - ✓ Fácil reprogramación, versatilidad característica de las computadoras actuales
 - => Compiladores, sistemas operativos, ...

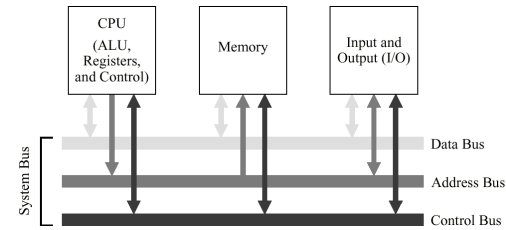
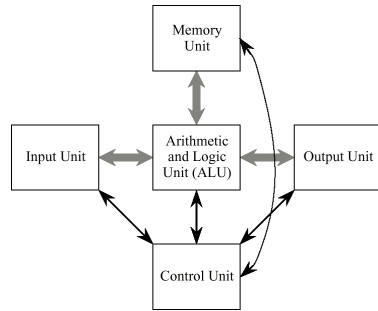
- ❖ Alternativamente la “arquitectura Harvard” (datos y programa almacenados en ubicaciones físicas diferentes) es aplicada ventajosamente en sistemas electrónicos pequeños
 - ✓ Dispositivos automáticos que cumplen una única función
 - ✓ Pequeño volumen de memoria de datos y de programa

Modelo bus de sistema

- ❖ La “arquitectura von Neumann” caracteriza a las computadoras actuales, pero se le ha incorporado un refinamiento: el **bus de sistema**
- ❖ Su propósito es reducir el número de interconexiones entre el CPU y sus subsistemas



Bus: {
▪ conjunto de cables individuales (1 bit/cable) agrupados por función
▪ por lo general bidireccionales



<h2>Modelo von Neumann</h2>	<h2>Modelo Bus de sistema</h2>
<ul style="list-style-type: none"> • Entradas y salidas separadas • Control y ALU separadas • <i>Intercomunicación:</i> Caminos separados 	<ul style="list-style-type: none"> • Unidad de entrada/salida • CPU= Control + ALU • <i>Intercomunicación:</i> Camino compartido = bus de sistema

El problema de la compatibilidad del software


Problema:

- Nueva máquina => desarrollar nuevo software

Solución:

- Compatibilidad “hacia arriba”
- Niveles de abstracción

Niveles de abstracción

- 
- *Usuario: no necesita saber programación*
 - *Lenguaje de alto nivel: datos e instrucciones del lenguaje, no importa cómo y dónde son manejados (el compilador se encarga)*
 - *Código de máquina (Assembler)*
 - *Diseño de lógica cableada o bien microprogramada*
 - *Unidades funcionales (registros, ALU ...)*
 - *Compuertas lógicas*
 - *Transistores, diodos, cables, etc.*

❖ **Esto permite generar productos con compatibilidad “hacia arriba”**

✓ **La línea IBM 360 fue la primera en garantizar compatibilidad hacia arriba en su código binario**

Niveles de abstracción

❖ Punto de vista del programador de computadoras

Arquitectura del set instrucciones (ISA)

- ✓ Conjunto de instrucciones
- ✓ Hardware accesible al programador
- ✓ El concepto ISA apareció con la familia *IBM-360*
- ✓ **No** tiene en cuenta ciclos de reloj, detalles de implementación del hardware, etc

❖ Punto de vista del diseñador del sistema

- ✓ Visión del conjunto (todos los niveles de abstracción)
- ✓ Optimización por velocidad, costos, cap. almacenamiento ...
- ✓ Soluciones de compromiso entre los distintos niveles de abstr.

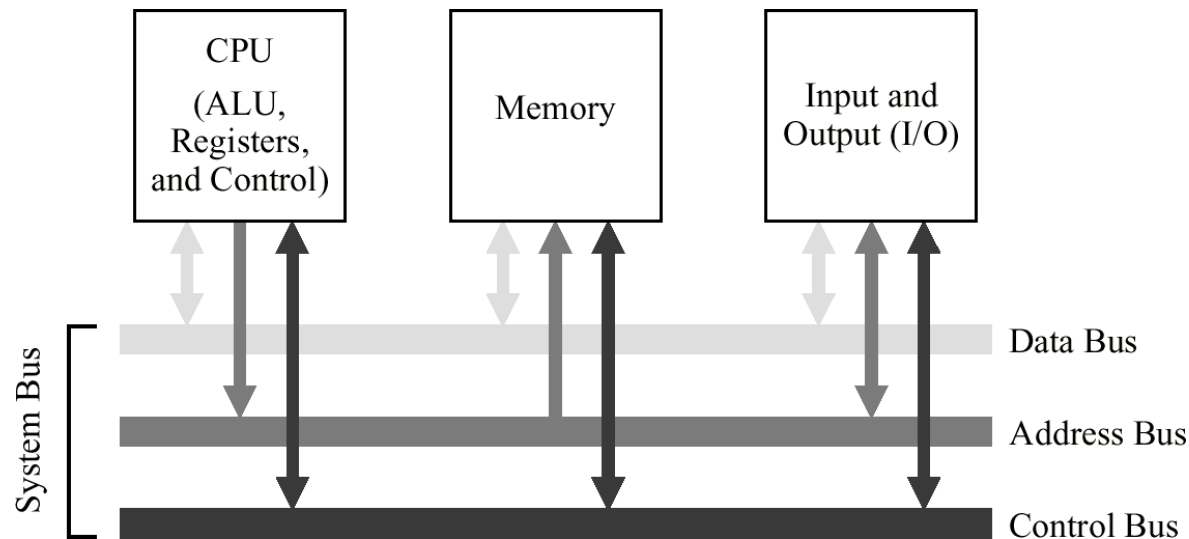
Programación a nivel código de máquina

- *Lenguajes de alto nivel son independientes del hardware*
- *Un compilador lo convierte en lenguaje de máquina*
- *Lenguaje Assembler es propio de cada procesador*
- *Lenguaje Assembler está compuesto por símbolos significativos para un ser humano (Add, jmp, mov)*
- *Un programa ensamblador convierte el lenguaje Assembler en código de máquina (serie de 0's y 1's)*

"Add r0, r1, r2" -> 0110101110101101

Ejecutar un programa bajo el modelo del bus de sistema

- *Programa (código máquina) es traído desde disco rígido a memoria*
- *CPU lee instrucciones y datos desde memoria*
- *Ejecuta cada instrucción en la secuencia programada y copiando resultados a memoria o E/S*



Memoria

Tamaños comunes para tipos de datos

Bit	0
Nibble	0110
Byte	10110000
16-bit word (halfword)	11001001 01000110
32-bit word	10110100 00110101 10011001 01011000
64-bit word (double)	01011000 01010101 10110000 11110011
	11001110 11101110 01111000 00110101
128-bit word (quad)	01011000 01010101 10110000 11110011
	11001110 11101110 01111000 00110101
	00001011 10100110 11110010 11100110
	10100100 01000100 10100101 01010001

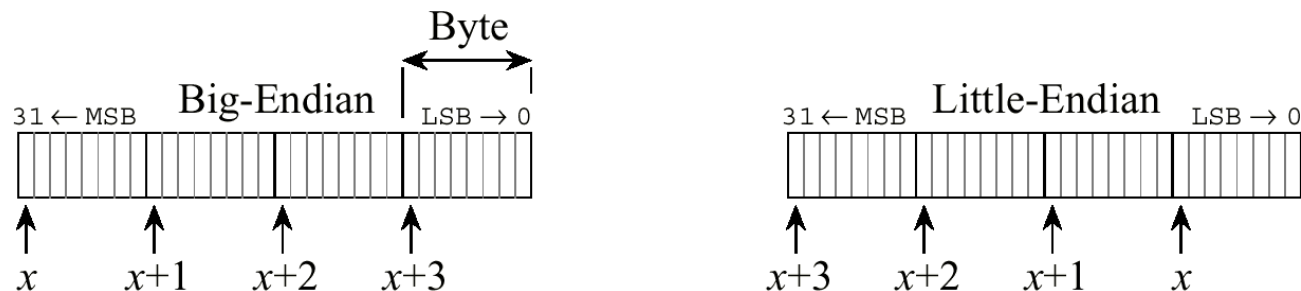
Memoria

- ✓ Forma una estructura de datos organizada tipo tabla
- ✓ Cada renglón de la tabla es identificado por su “dirección”
- ✓ Cada dato es agrupado físicamente de a 8 bits (=1 byte)
- ✓ Palabras de mas de 8 bits son guardadas como serie de bytes
- ✓ Los procesadores en general tienen instrucciones para acceder directamente a palabras de 1 byte o más

Formatos

Little-Endian y Big-Endian

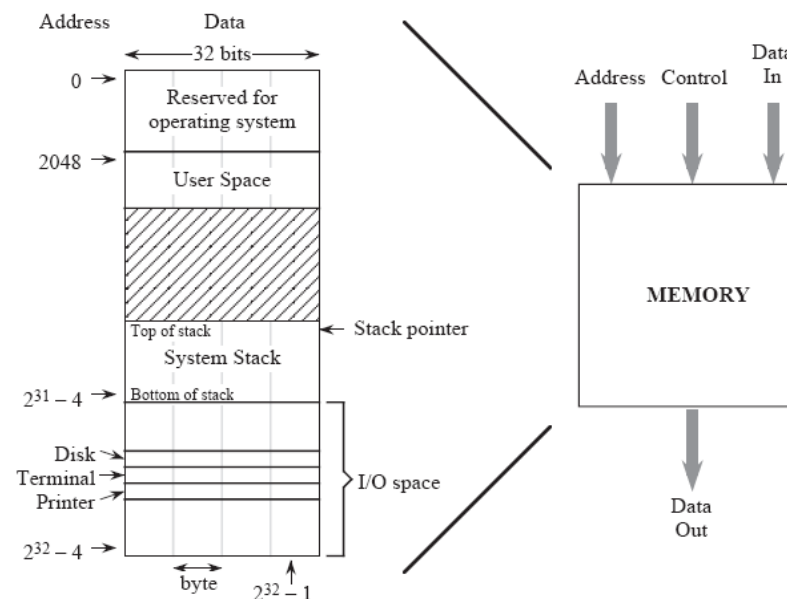
- La memoria es direccionable por bytes
- Palabras multi-byte
 - El byte menos significativo en la dirección más baja
Little-Endian
 - El byte menos significativo en la dirección más alta
Big-Endian
- La dirección de la palabra multibyte es la dirección más baja



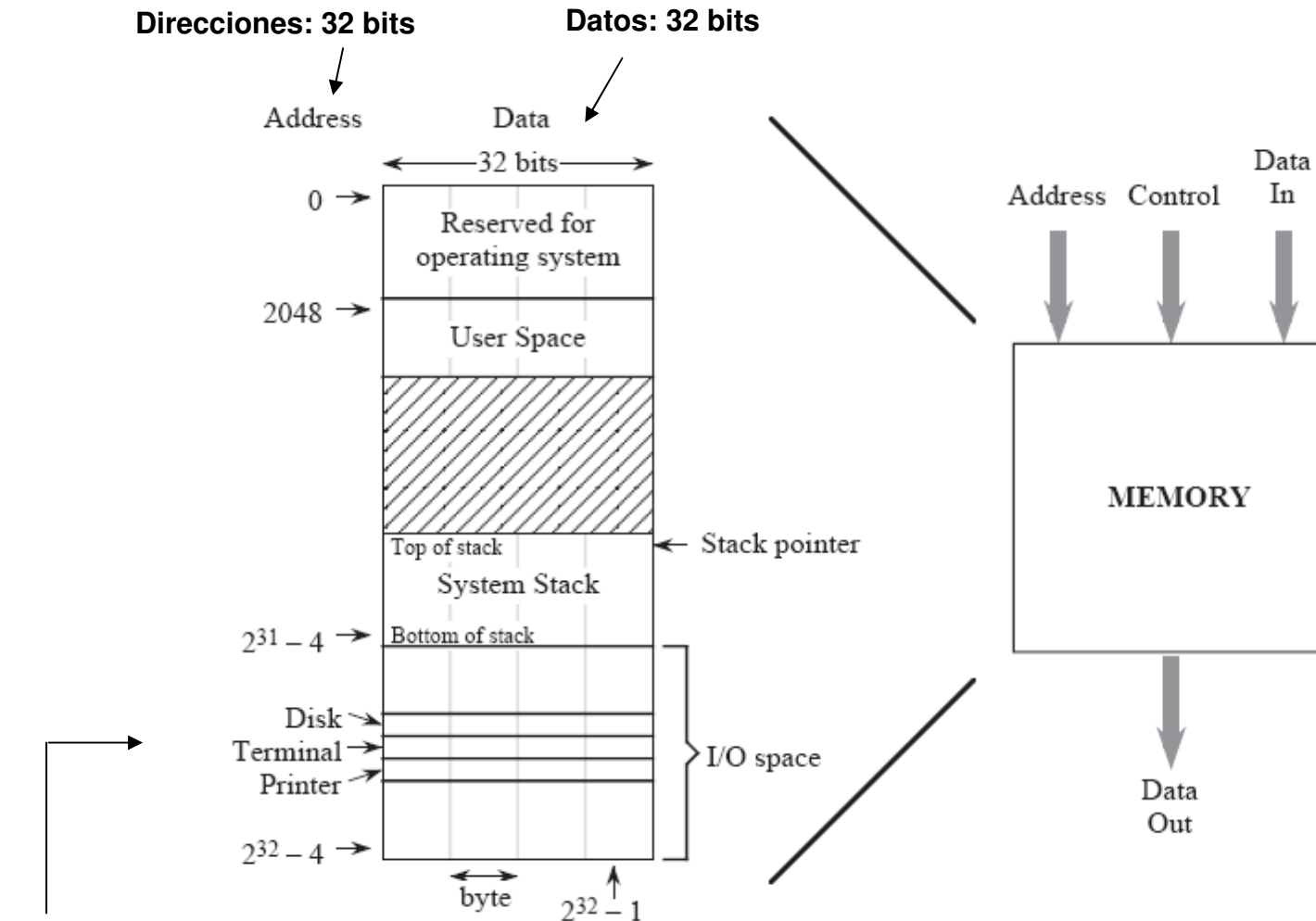
x : dirección de memoria de la palabra de 4 bytes

Mapa de Memoria

- . Grafica la forma en que se organiza el espacio de memoria
- . El espacio de memoria disponible depende del procesador
- . Una sistema (computadora) tiene un mapa de memoria específico
- . Dos sistemas basadas en el mismo procesador no tienen necesariamente el mismo mapa de memoria



Mapa de Memoria



Dispositivos de entrada/salida
mapeados en memoria

Relacionar con diagrama
de bus de sistema

¿En base a qué se define el rango
de direcciones?

Qué puede hacer una
computadora?

Qué puede hacer una computadora?

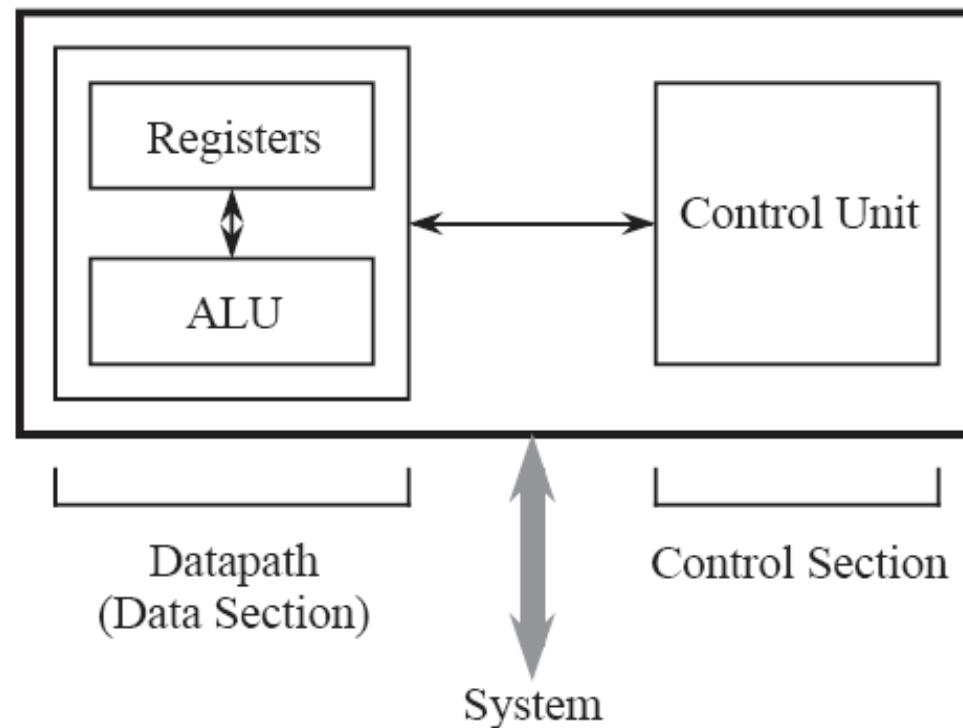
- Las tareas que implementa pueden ser muy complejas
- Los modos de operación internos son sorprendentemente limitados

Esencialmente, lo que puede hacer es:

- Ejecutar operaciones en una secuencia ordenada
- Existen dos tipos básicos de operaciones:
 - Mover datos
 - Operaciones aritméticas simples y oper. lógicas

Unidad Central de Proceso CPU

- ❖ Sección de control
- ❖ Sección de datos (“camino de datos” o “datapath”)



Sección de control

- *Función:* Controlar la ejecución de las instrucciones de programa que están almacenadas en memoria principal
- *Principales registros:*
 - ✦ **PC** (Contador de programa)
 - ✦ **IR** (Registro de instrucciones)
- *Ejecutar una instrucción de progr. significa:*
 - 1) Buscar en memoria la próxima instrucción a ser ejecutada
 - 2) Decodificar el código de operación de esa instrucción
 - 3) Traer operandos desde memoria principal, si los hubiera
 - 4) Ejecutar la instrucción y guardar los resultados
 - 5) Volver a (1)

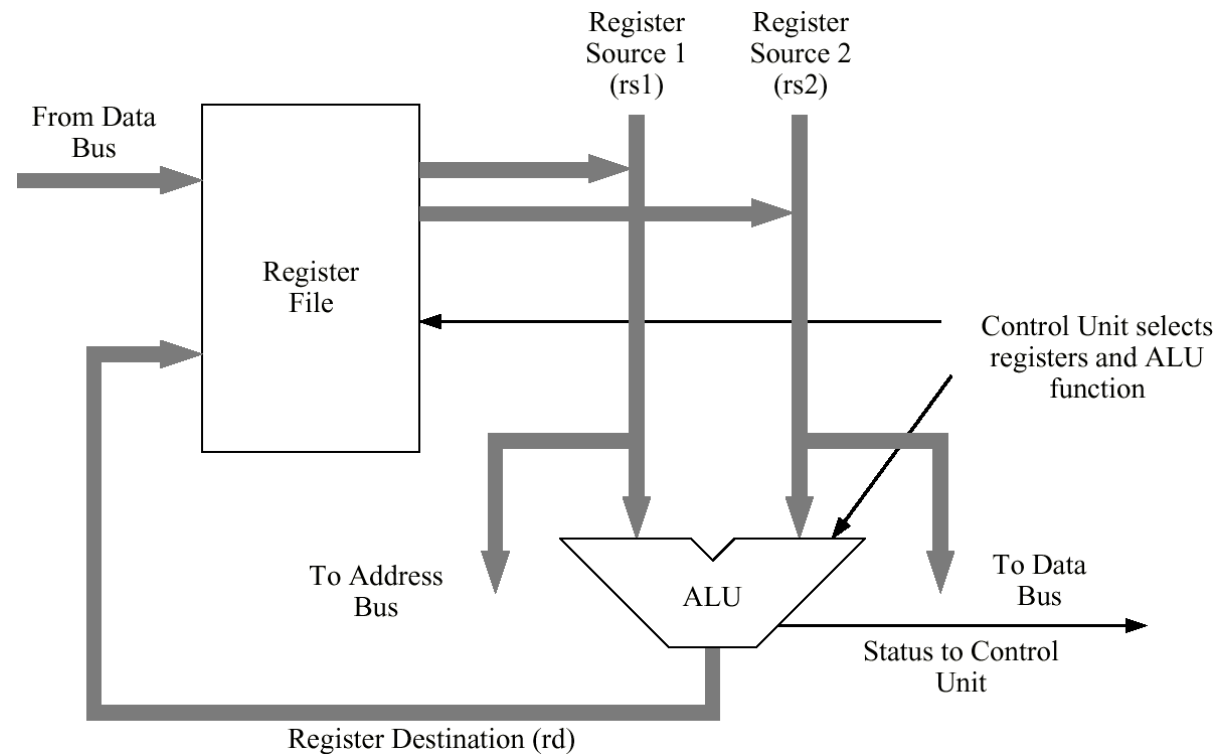


(“Ciclo de búsqueda-ejecución”, “ciclo de fetch”, “ciclo de “fetch-execute”)

Sección de datos

➤ *Función:* Realizar la tarea que involucra cada instrucción. Actúa bajo el control de la sección control. Para ello cuenta con:

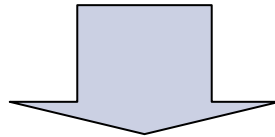
- ❖ Archivo de registros
- ❖ Unidad aritmético-lógica (ALU)



Sección de datos

Archivo de registros

- ❖ Es memoria interna al CPU organizada como RAM
- ❖ Bus de direcciones (interno y de pocos bits)
- ❖ Cada registro está implementado con circuitos muy rápidos



Programas con uso intensivo de registros son más rápidos
que los que hacen uso intensivo de memoria principal

Sección de datos

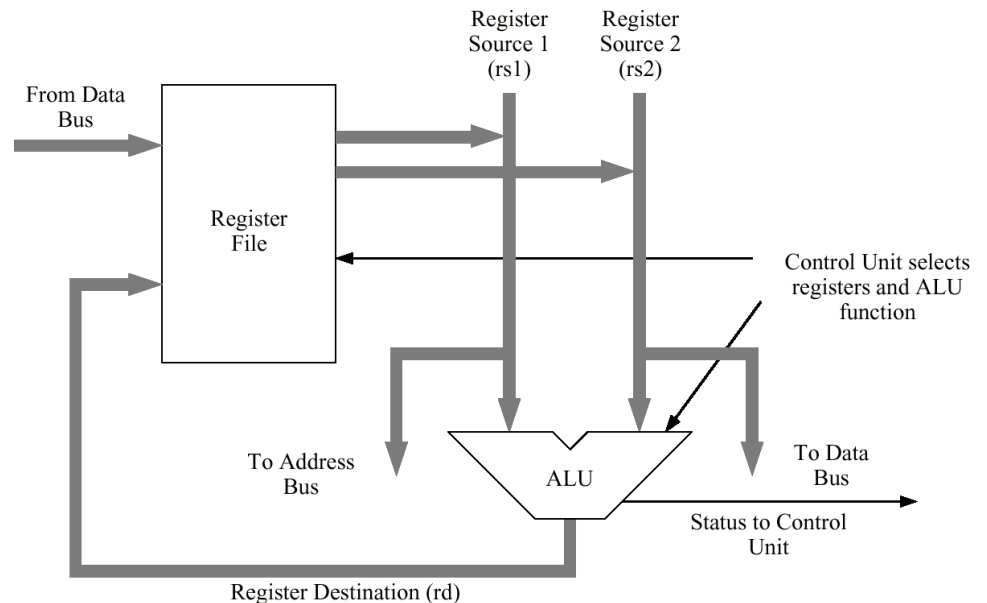
Buses

Internos a la CPU

- ❖ Entrada ALU: 2 Buses de Datos (rs1 y rs2)
- ❖ Resultado de ALU hacia registros: 1 Bus de datos (rd)

Conectados con el bus de sistema

- ❖ Desde DATA BUS al archivo de registros
- ❖ Hacia DATA BUS (rs2)
- ❖ Hacia ADDRESS BUS (rs1)



Set de instrucciones

➤ Características típicas

- ❖ Tamaño de las instrucciones
- ❖ Tipo de operaciones admitidas
- ❖ Tipo de operandos (ubicación y tamaño)
- ❖ Tipo de resultados

➤ Compatibilidad del software

- ❖ Programa compilado para PC no anda en PowerPC (Mac / IBM)
- ❖ Lenguajes de alto nivel son idénticos pero requieren recompilación
- ❖ Programa compilado para Mac no anda en IBM pSeries ¿?