Organización del Computador 1

Máquina de von Neumann

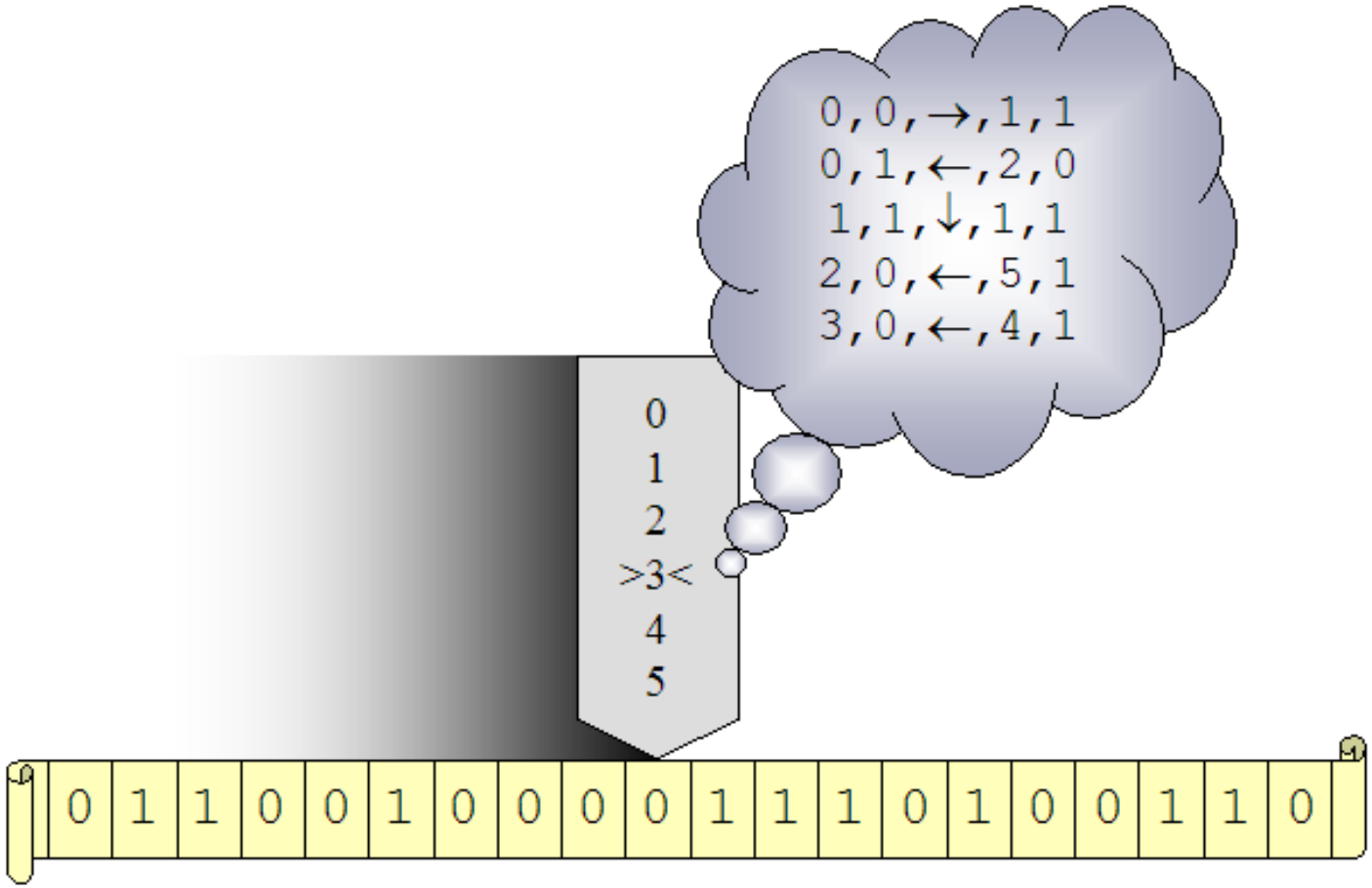
Jerarquía de Niveles

Inicios de la computación

Turing y Church sientan las bases teóricas de la computación

- Máquina de Turing
  - Máquina teórica compuesta por una cinta y una cabeza que puede leer y grabar símbolos en ella.
  - Un conjunto finito de estados
  - Un programa “cableado” del tipo (condición, acción)
- Church: Lambda calculo
  - Resultado equivalente para probar computabilidad
  - Base de los lenguajes funcionales

Una máquina de Turing



Inicios de la computación

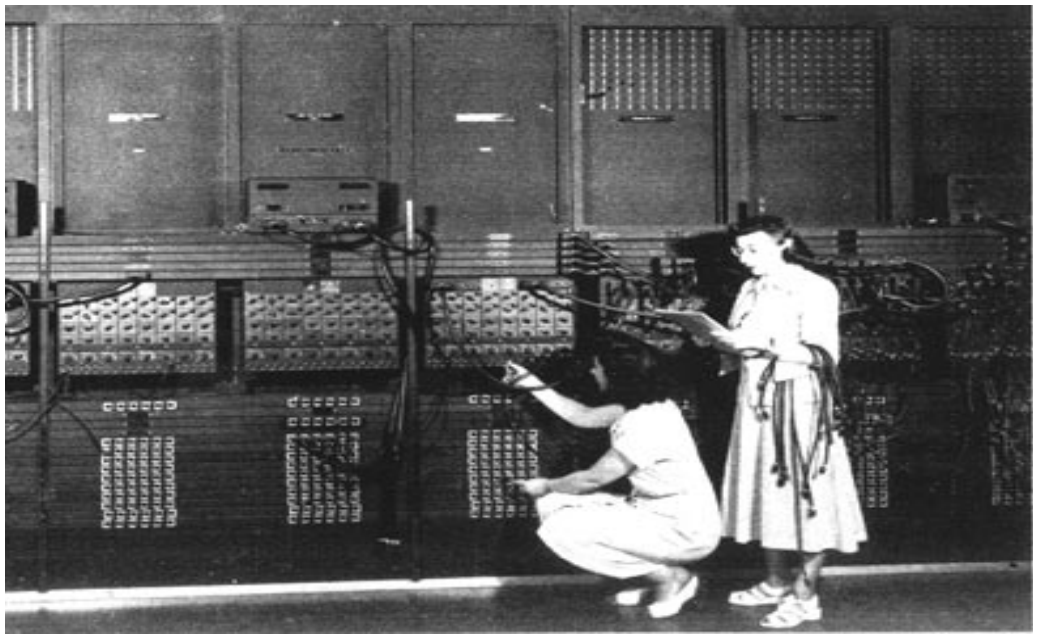
Turing y Church sientan las bases teóricas d
computación

- Maquina de Turing
  - Máquina teórica compuesta por una cinta y un
    cabeza que puede leer y grabar símbolos en e
  - Un conjunto finito de estados
  - Un programa “cableado” del tipo (condición, acción)

- Máquina universal de Turing
  - Máquina (teórica) capaz de simular el comportamiento **de cualquier maquina** (de Turing) a partir de un programa ingresado en la cinta

El modelo de von Neumann

- Antes: programar era conectar cables…
- Hacer programas era mas una cuestión de ingeniería electrónica
- Cada vez que había que calcular algo distinto había que reconectar todo.
- Imaginen eso !



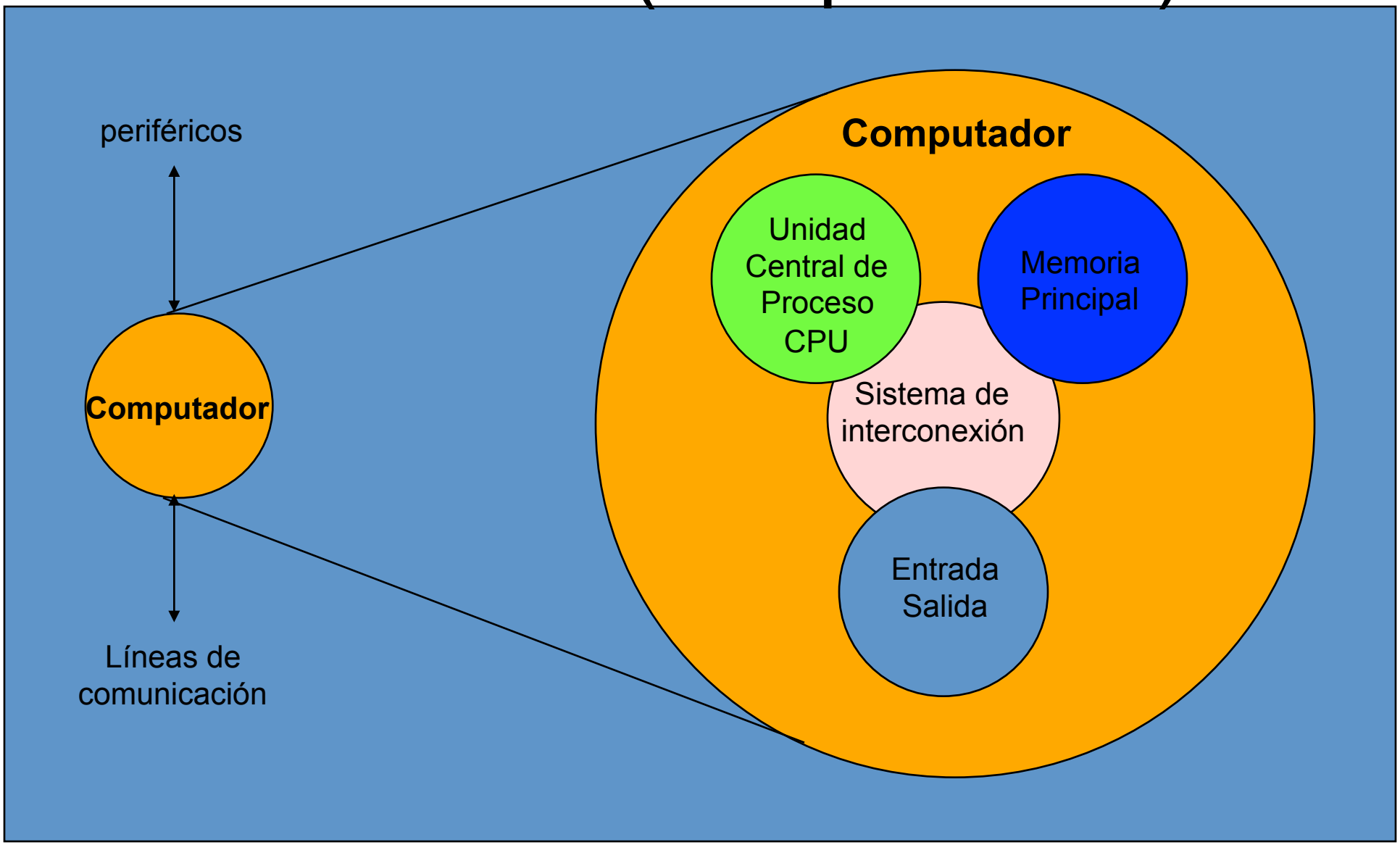
John Von Neumann

- 1903 – 1957
- Matemático
- Publicó y publicitó la idea de **programa almacenado en memoria**
- 1945: “Primer Borrador de un Reporte sobre la EDVAC”

von Neumann

- Los datos y programas se almacenan en una misma memoria de lectura-escritura
- Los contenidos de esta memoria se direccionan indicando su posición sin importar su tipo
- Ejecución en secuencia (salvo que se indique lo contrario)
- Representación Binaria

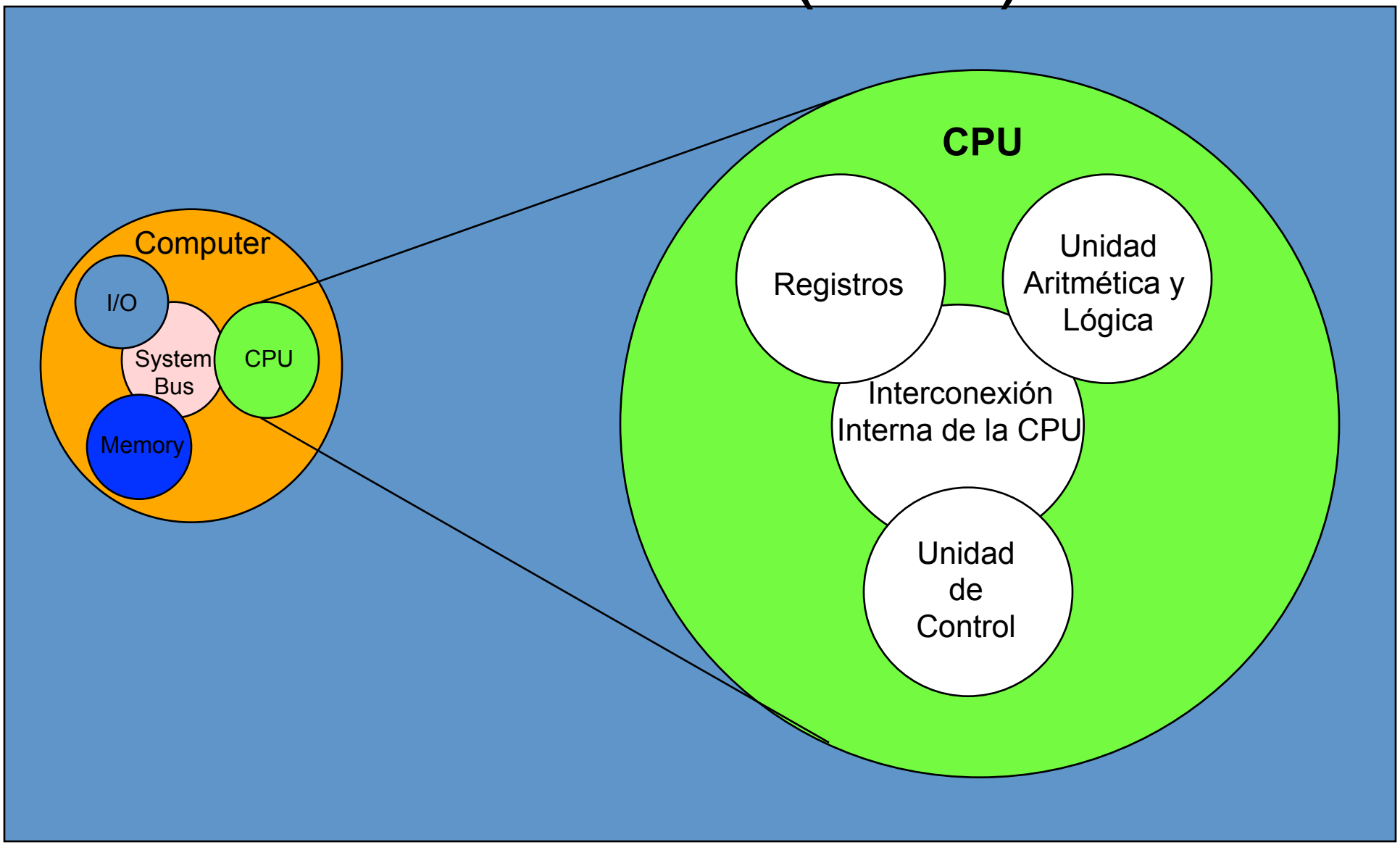
Estructura (computadora)



Características principales

- 3 componentes principales:
  - CPU:
    - Unidad de Control, Unidad aritmético lógica (ALU), Registros
  - Memoria principal:
    - Almacena programas y datos
  - Sistema de Entrada/Salida
- Procesamiento secuencial de instrucciones
- Datos binarios
- Un sistema de interconexión
  - Conecta la memoria y unidad de control
  - Fuerza la alternación entre ciclos de lectura y ejecución

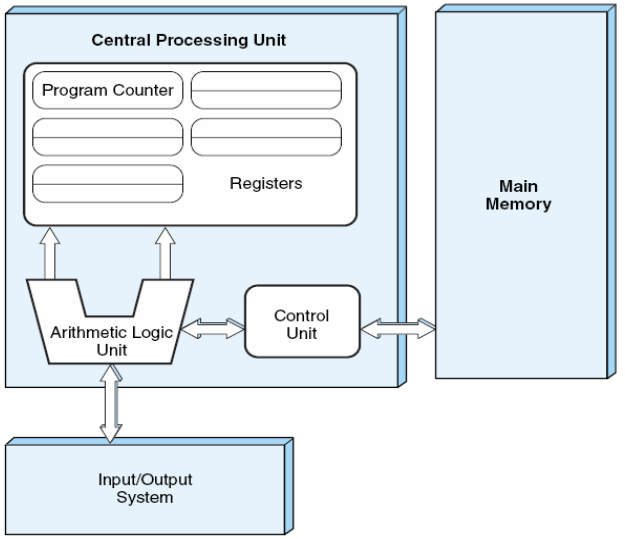
Estructura (CPU)



CPU

Unidad de Control (UC)

- Controla todos los componentes
- Interpreta instrucciones
  - Decodifica y Ejecuta instrucciones. Transforma instrucciones en órdenes a otros componentes
  - Puede ser programada por hardware (cableada) y “microprogramada” (varias microinstrucciones por instrucción)



Unidad Aritmético Lógica (ALU)

- Realiza operaciones matemáticas y lógicas
  - Sumas, restas, multiplicaciones
  - And, Or, Xor
  - Corrimientos

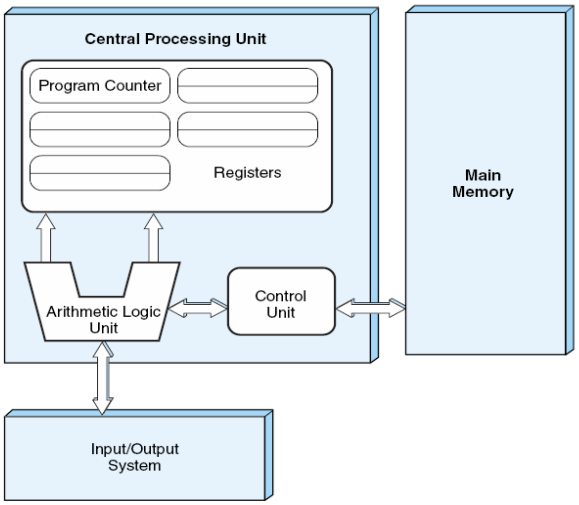
CPU

Registros

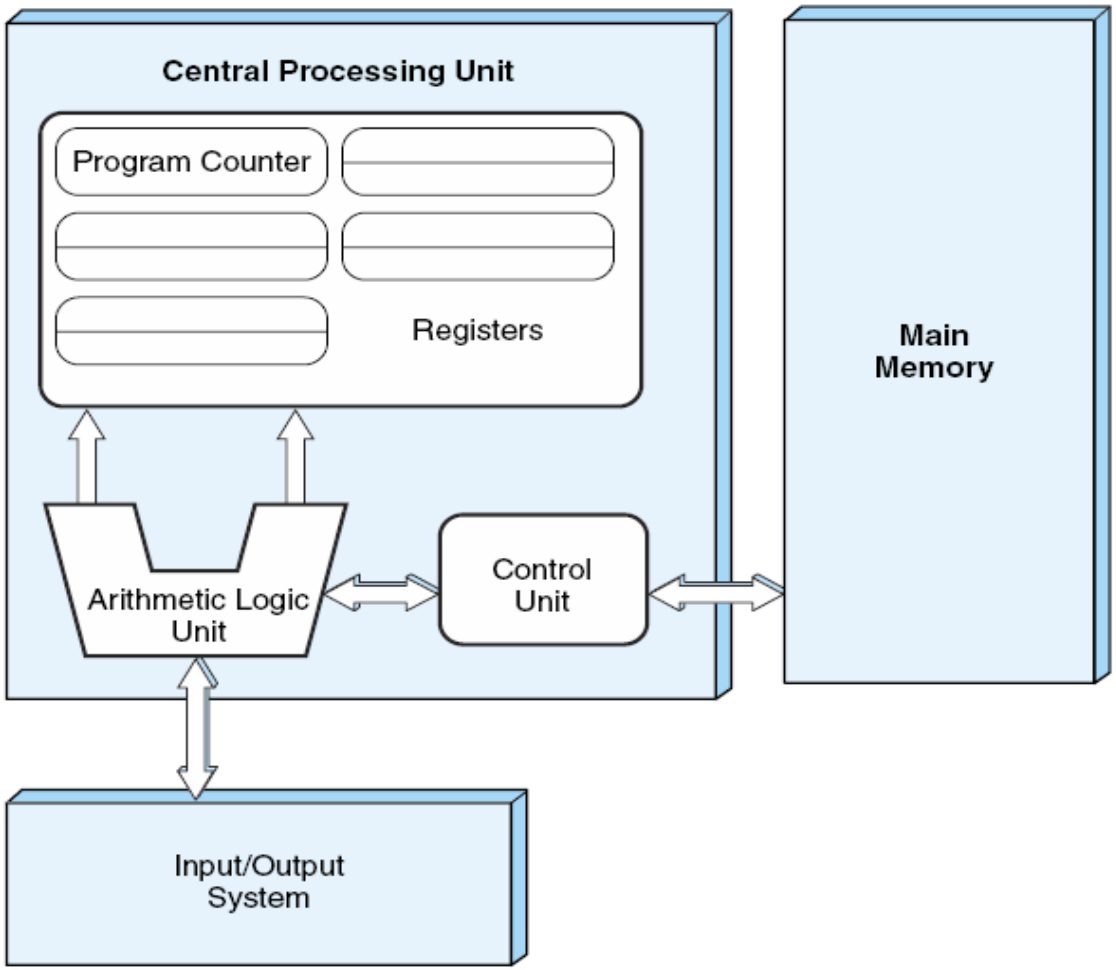
- Almacenan datos binarios, acceso rápido
- De tamaño fijo
- De propósito general (programas) o específicos (acumulador, program counter, puntero a memoria, etc.)

DataPath

- Red interna que comunica la UC con las otras unidades y registros
- Mueve datos entre los diferentes componentes
- Controlada por un reloj.

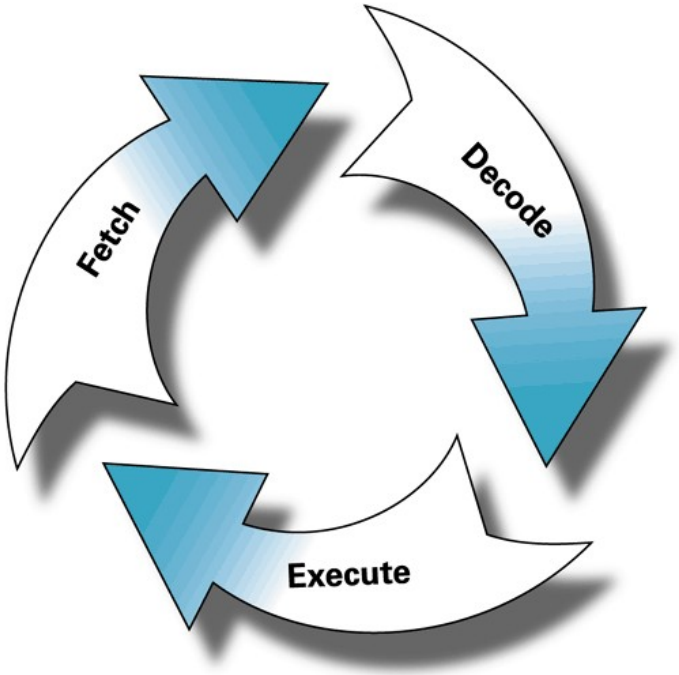


Estructura de una máquina von Neumann



Ciclo de instrucción

1. Recuperar la siguiente instrucción desde memoria (apuntada por el *program counter*) y luego incrementar el *program counter*.

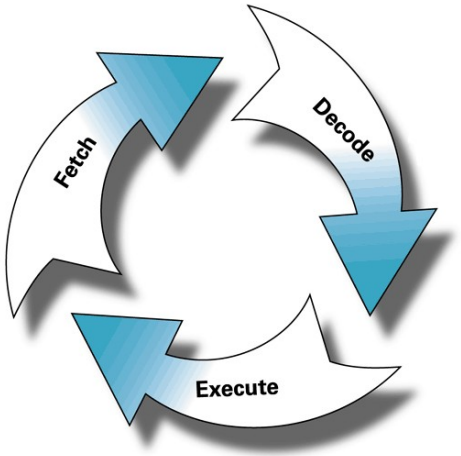


2. Decodificar el patrón de bits en el registro de instrucción IR

3. Ejecutar la instrucción indicada en el registro de instrucción IR

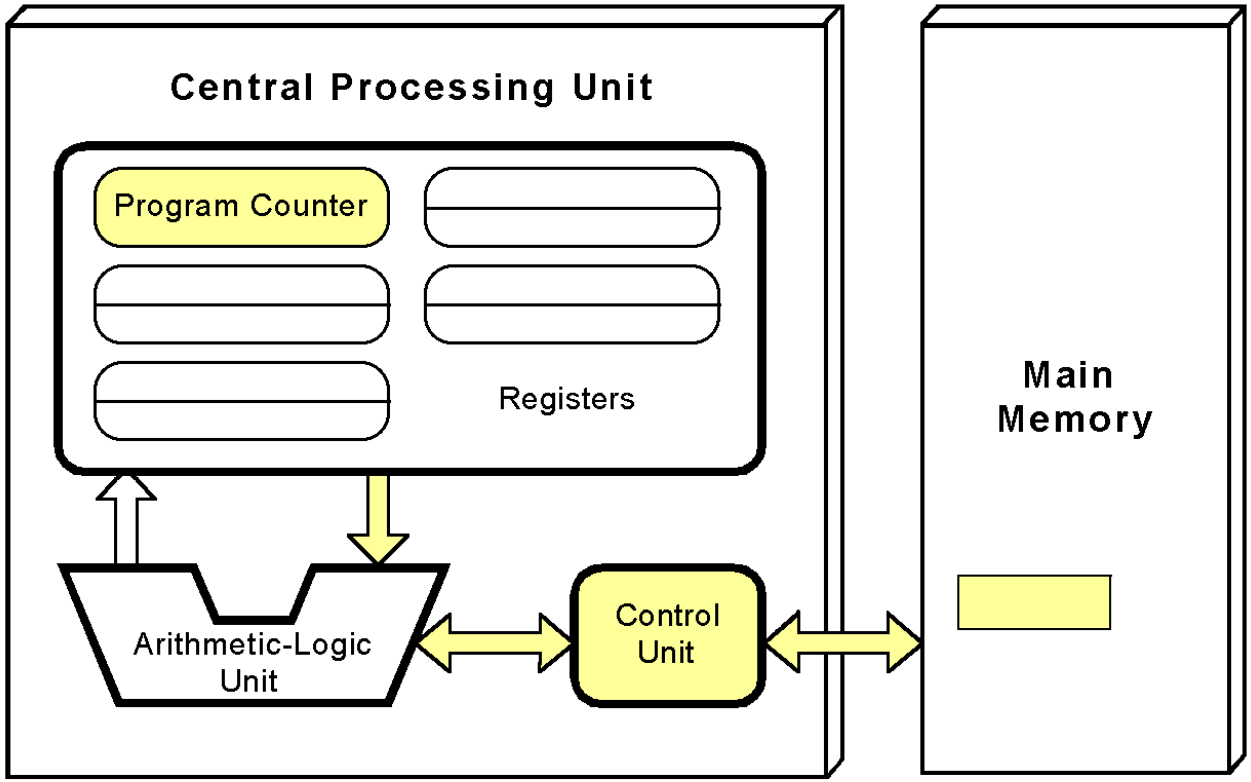
Ciclo de Ejecución

1. UC obtiene la próxima instrucción de memoria (usando el registro PC)
2. Se incrementa el PC
3. La instrucción es decodificada a un lenguaje que entiende la ALU
4. Obtiene de memoria los operandos requeridos por la operación
5. La ALU ejecuta y deja los resultados en registros o en memoria
6. Repetir paso 1



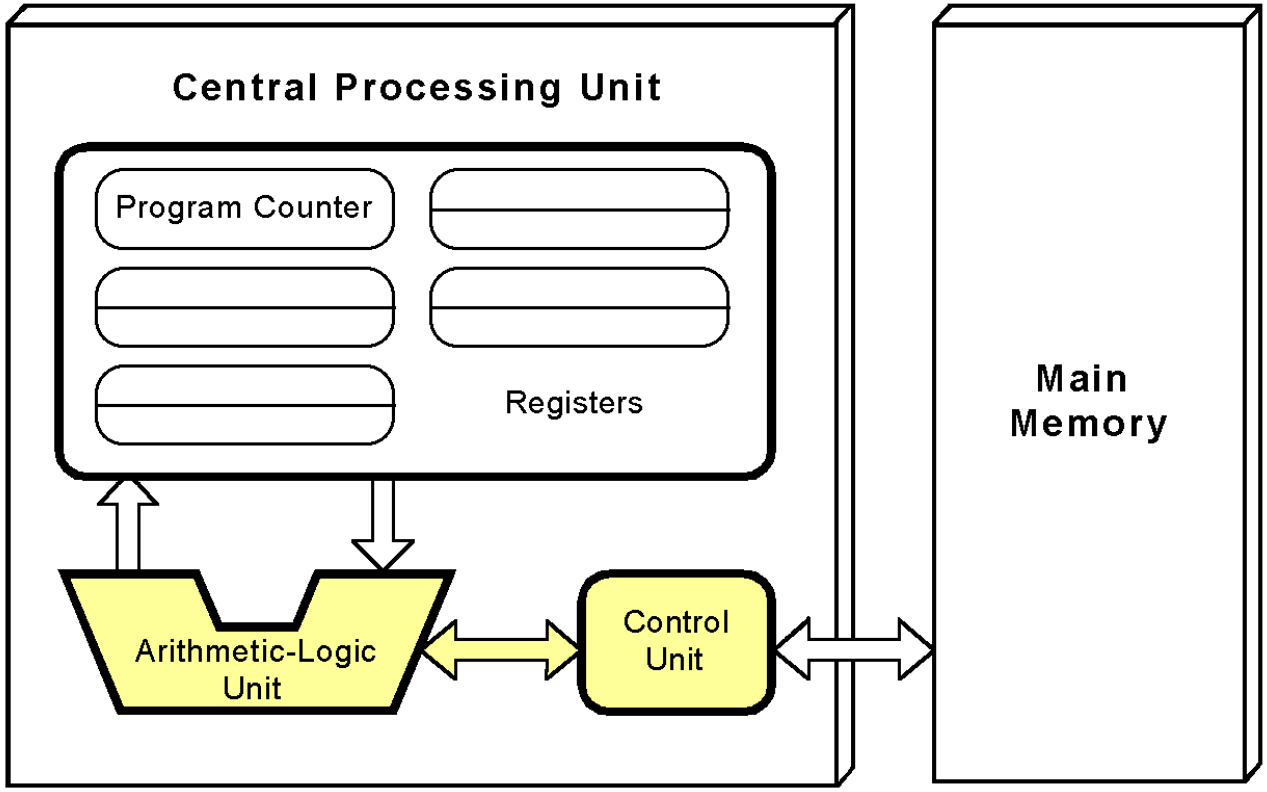
Ciclo de ejecución

- **La unidad de control levanta la próxima de memoria usando el "contador de programa" (o RPI) que dice en que dirección esta la próxima instrucción.**



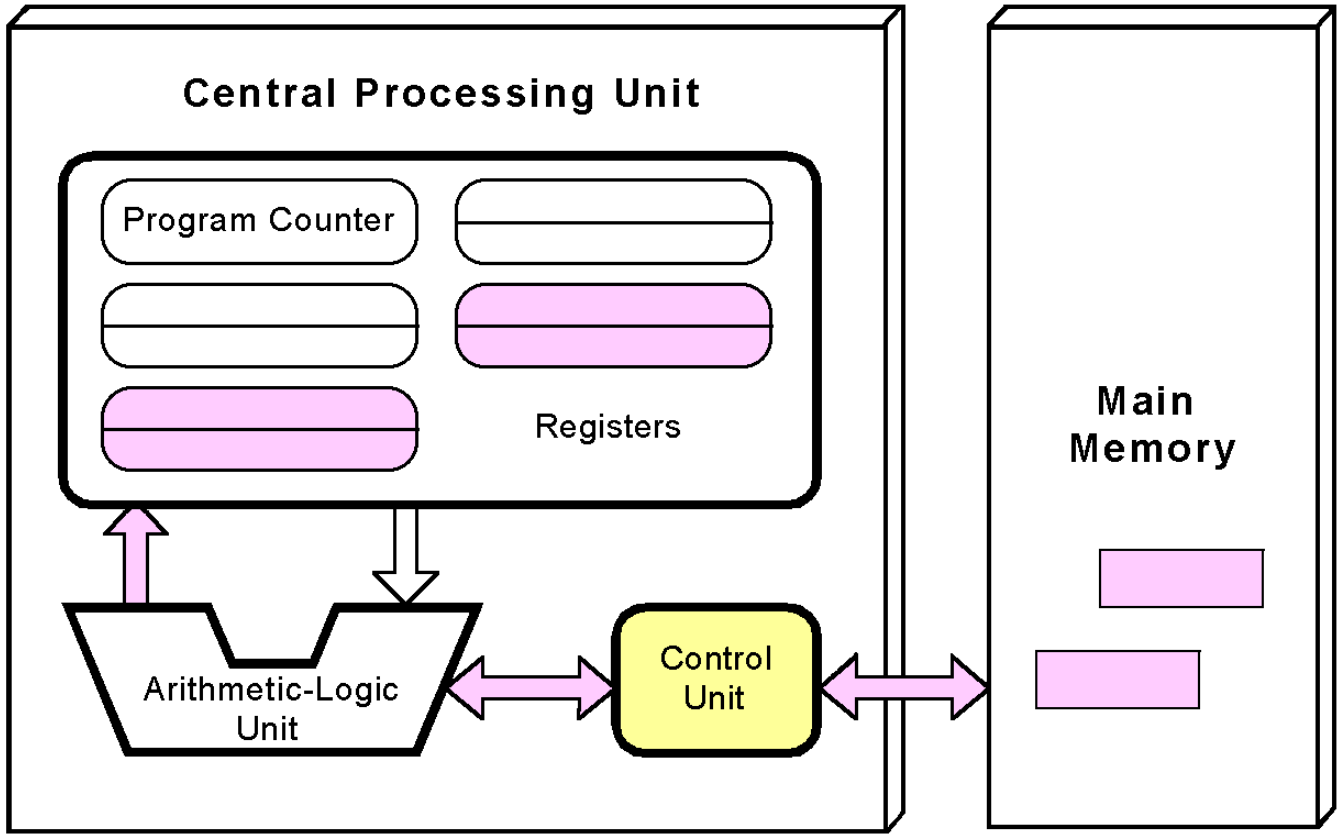
Ciclo de ejecución

- **La instrucción es decodificada a un lenguaje que entiende la ALU (unidad aritmética lógica).**



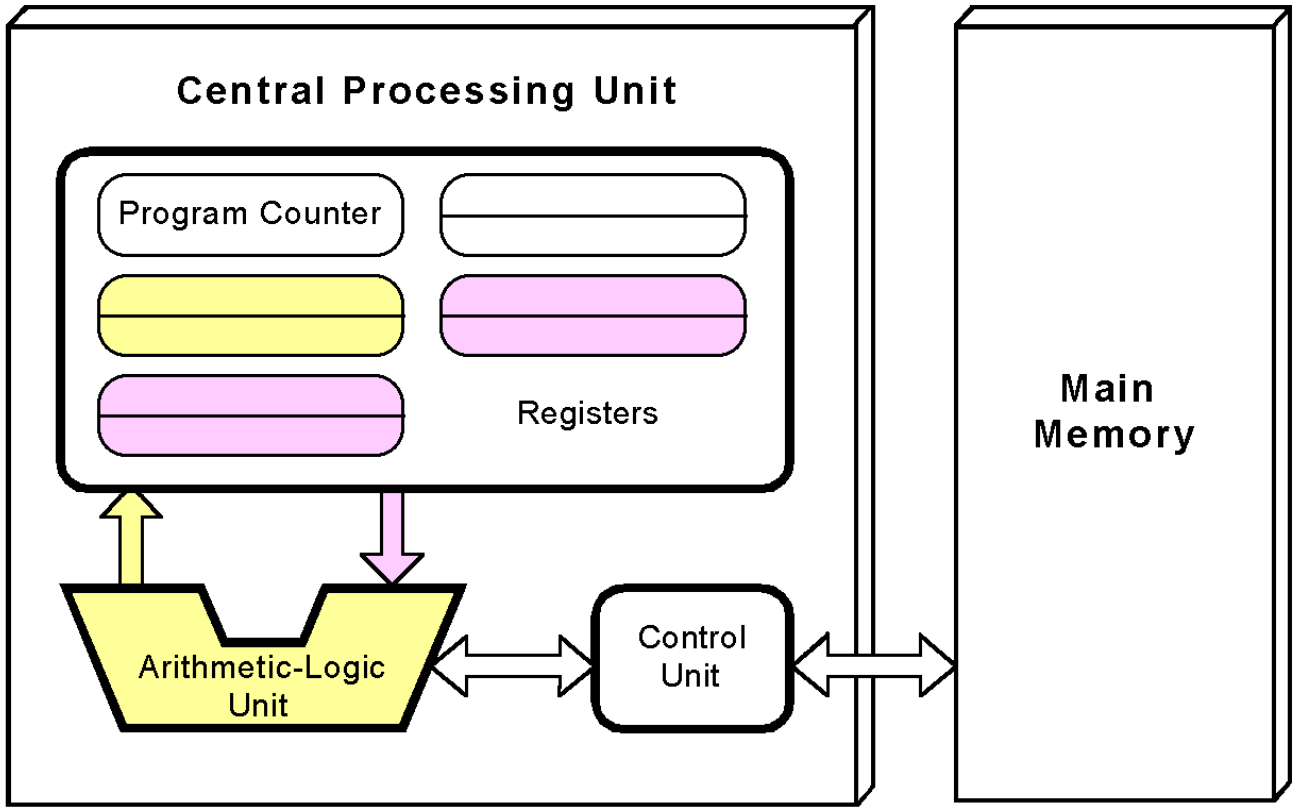
Ciclo de ejecución

- **Cada operando requerido para ejecutar es levantado de la memoria principal y ubicado en registros dentro de la CPU.**



Ciclo de ejecución

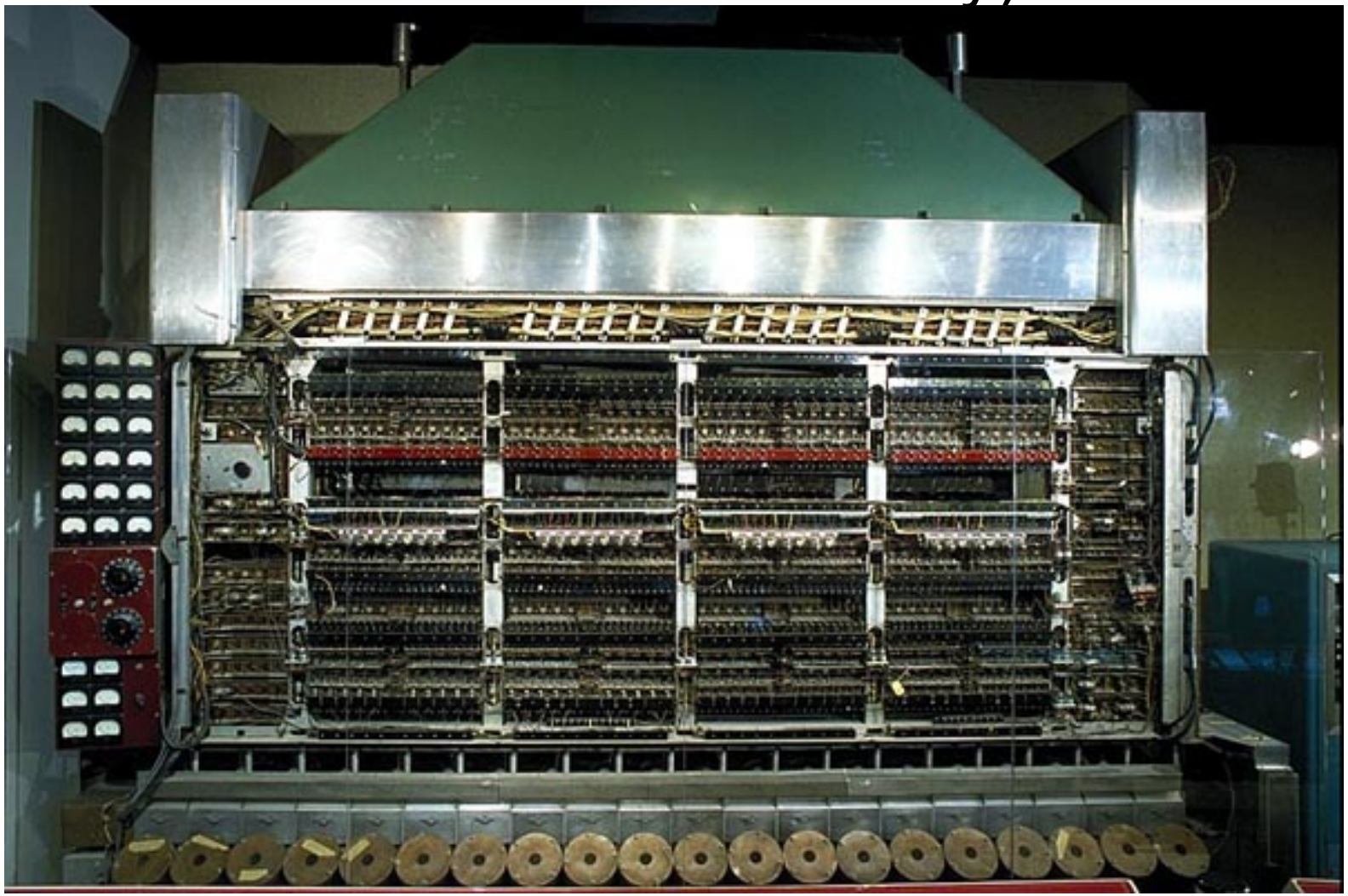
- **La ALU ejecuta la instrucción y coloca los resultados en registros o en memoria.**



Tipos de Operaciones

- Procesador-memoria
  - Transferencia de datos entre la CPU y la memoria
- Procesador-E/S
  - Transferencia de datos entre la CPU y un modulo de E/S
- Procesamiento de datos
  - Alguna operación aritmética o lógica sobre los datos
- Control
  - Alteración de la secuencia de operaciones
  - Ej.: jump

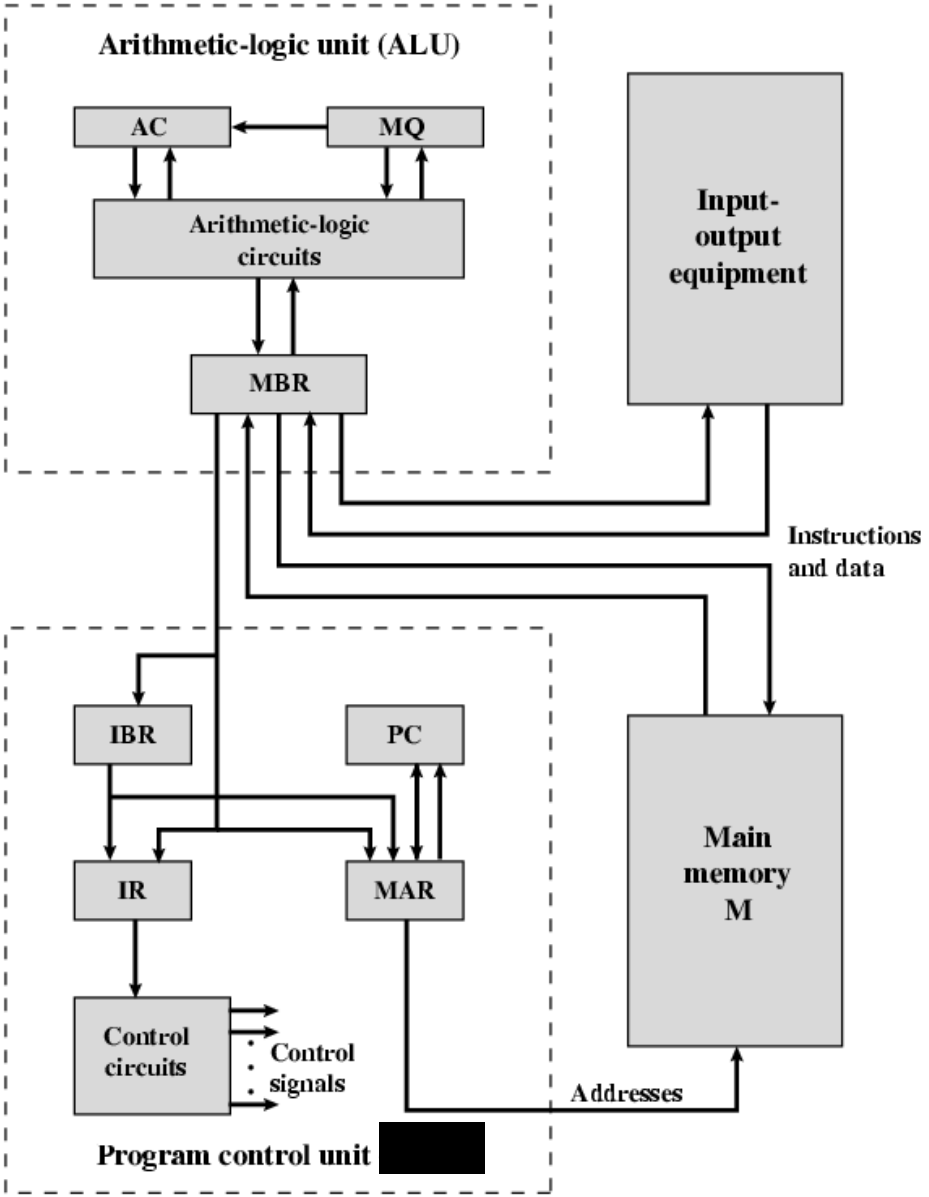
La IAS (Institute for Advance Study, Princeton University)



Estructura de la IAS

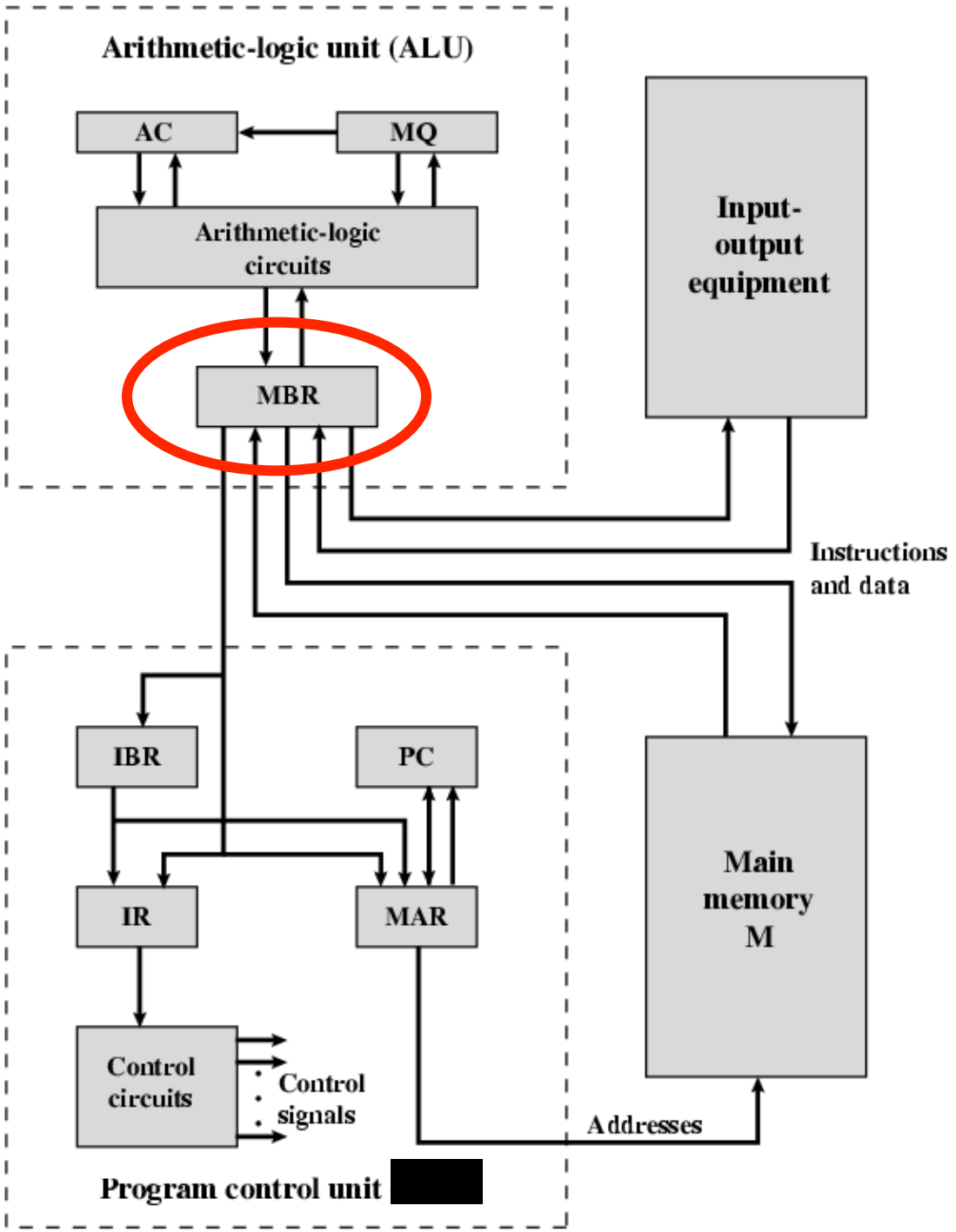
Registros

- Memory Buffer Register
- Memory Address Register
- Instruction Register
- Instruction Buffer Register
- Program Counter
- Accumulator
- Multiplier Quotient



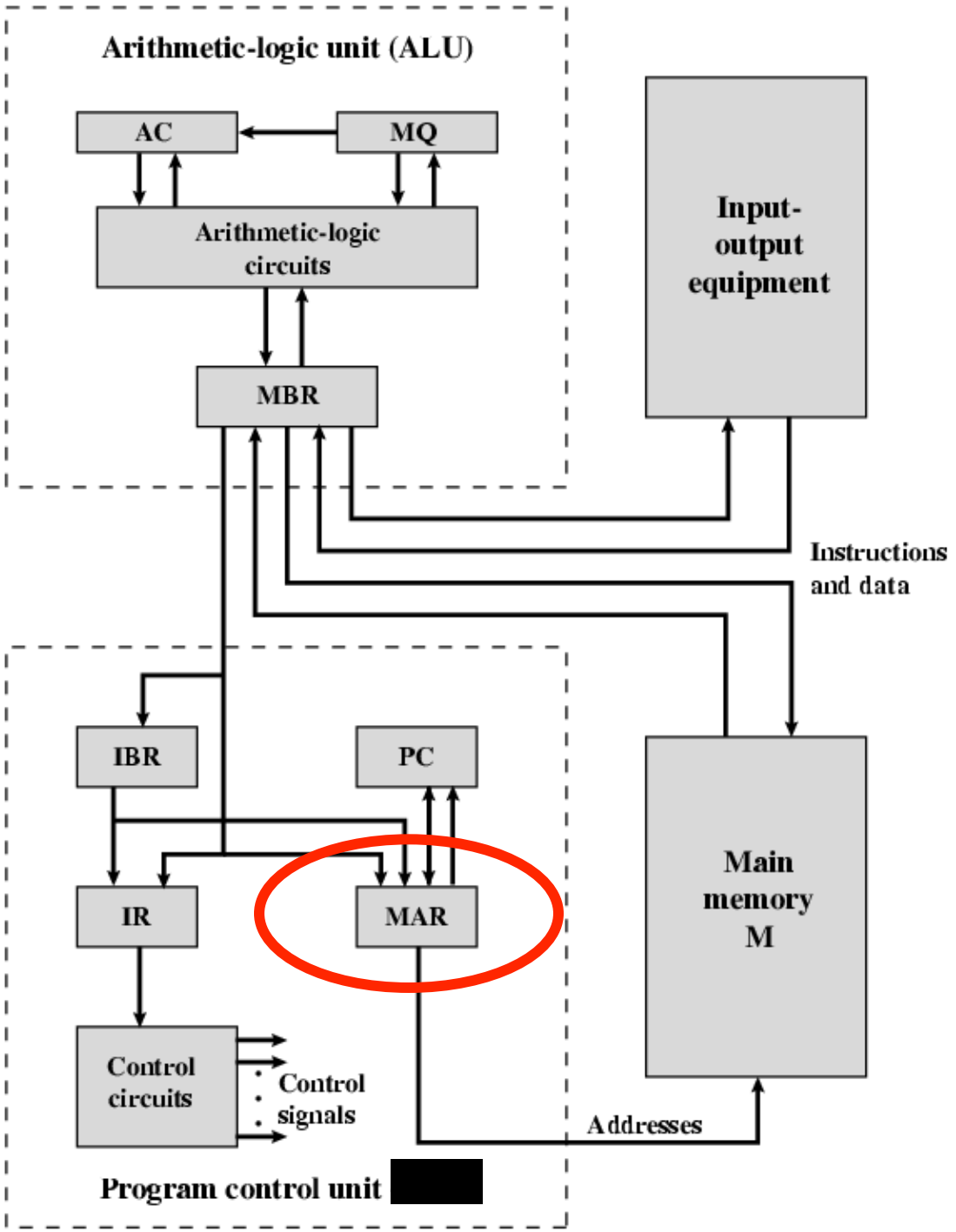
MBR:
Memory Buffer Register

Contiene una palabra que debe ser almacenada en la memoria, o es usado para recibir una palabra procedente de la memoria.



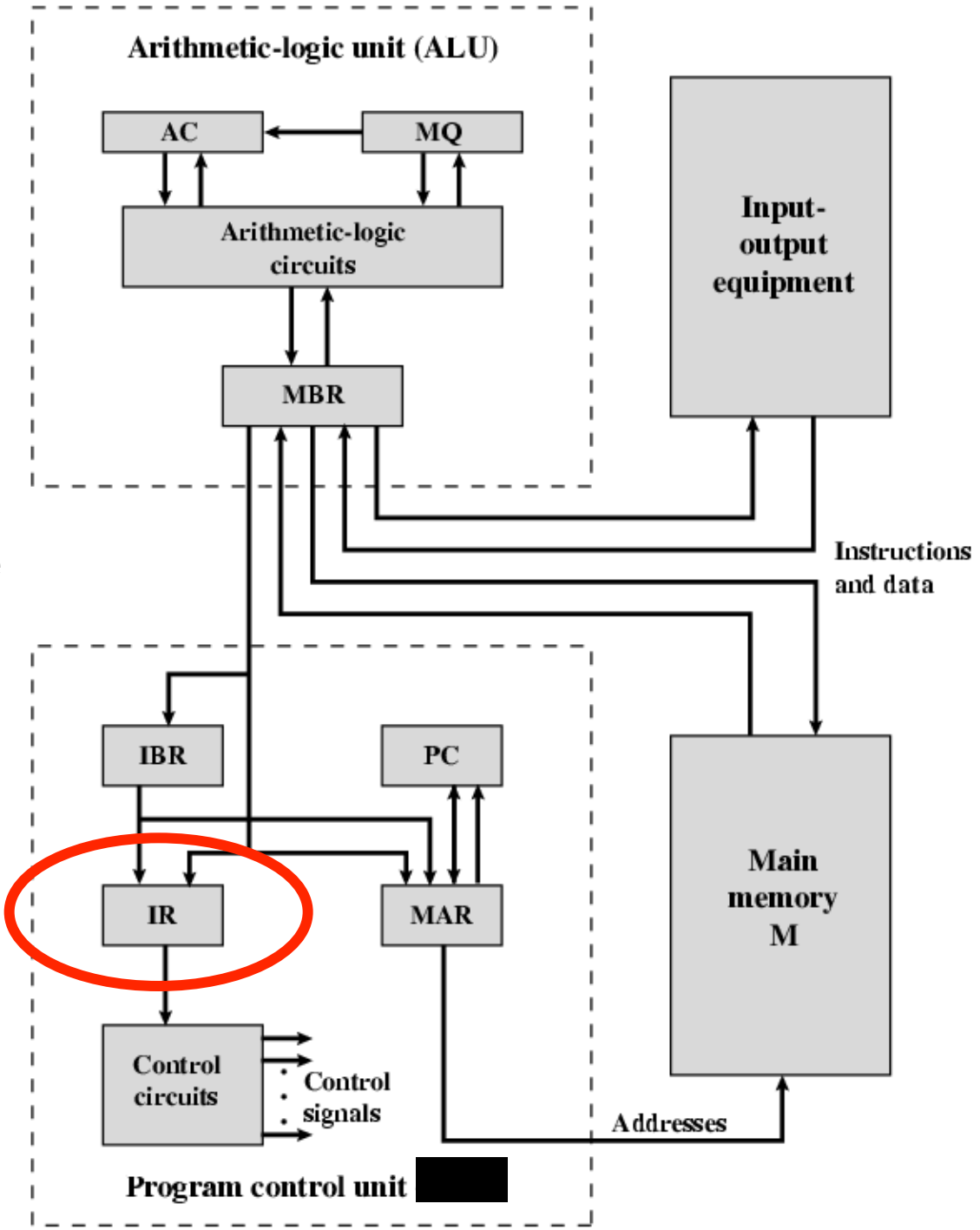
MAR:
Memory Adress Register

Especifica la dirección en memoria de la palabra que va a ser escrita o leída en MBR.



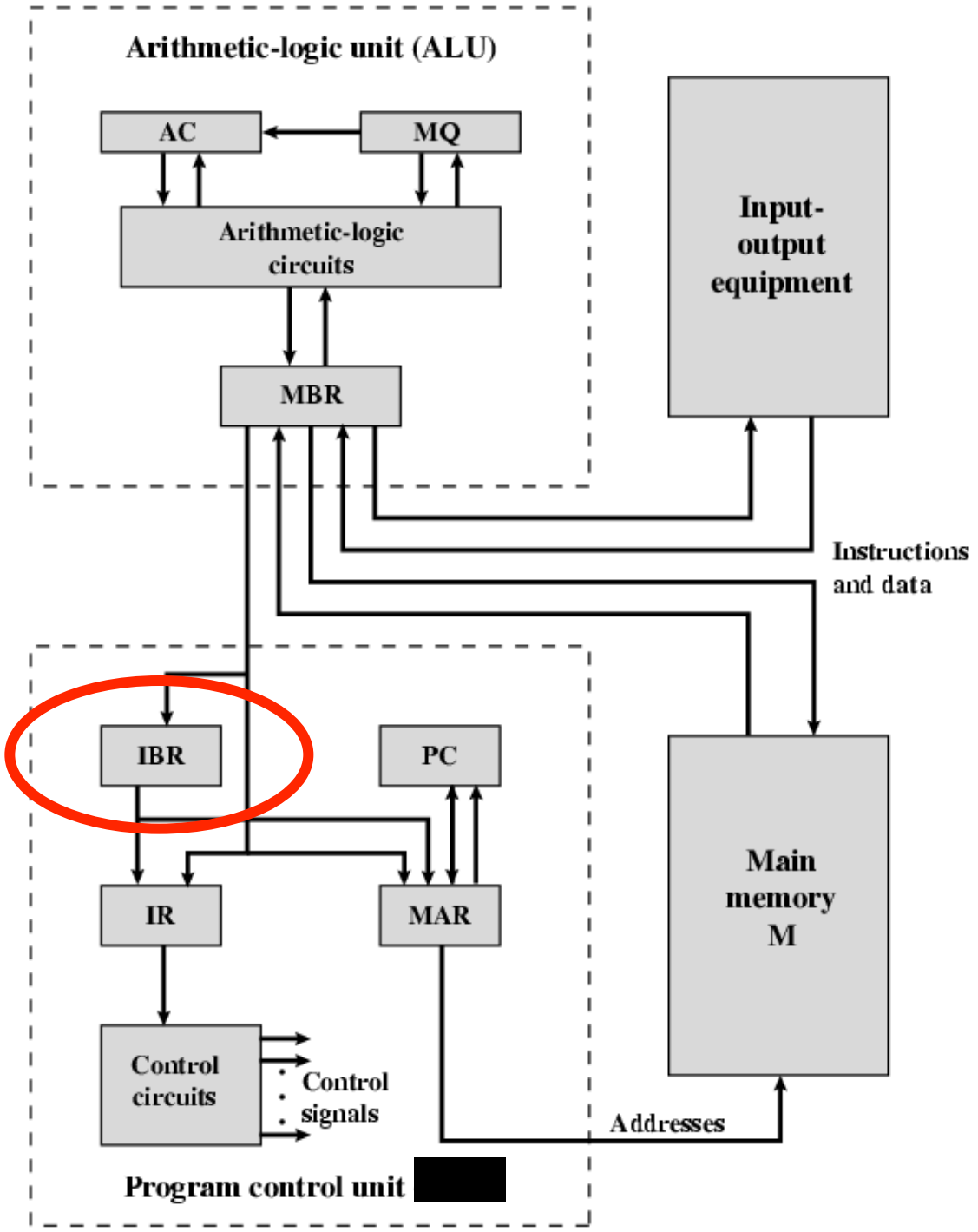
IR:
Instruction Register

Contiene los 8 bits del código de operación de la instrucción que se va a ejecutar.



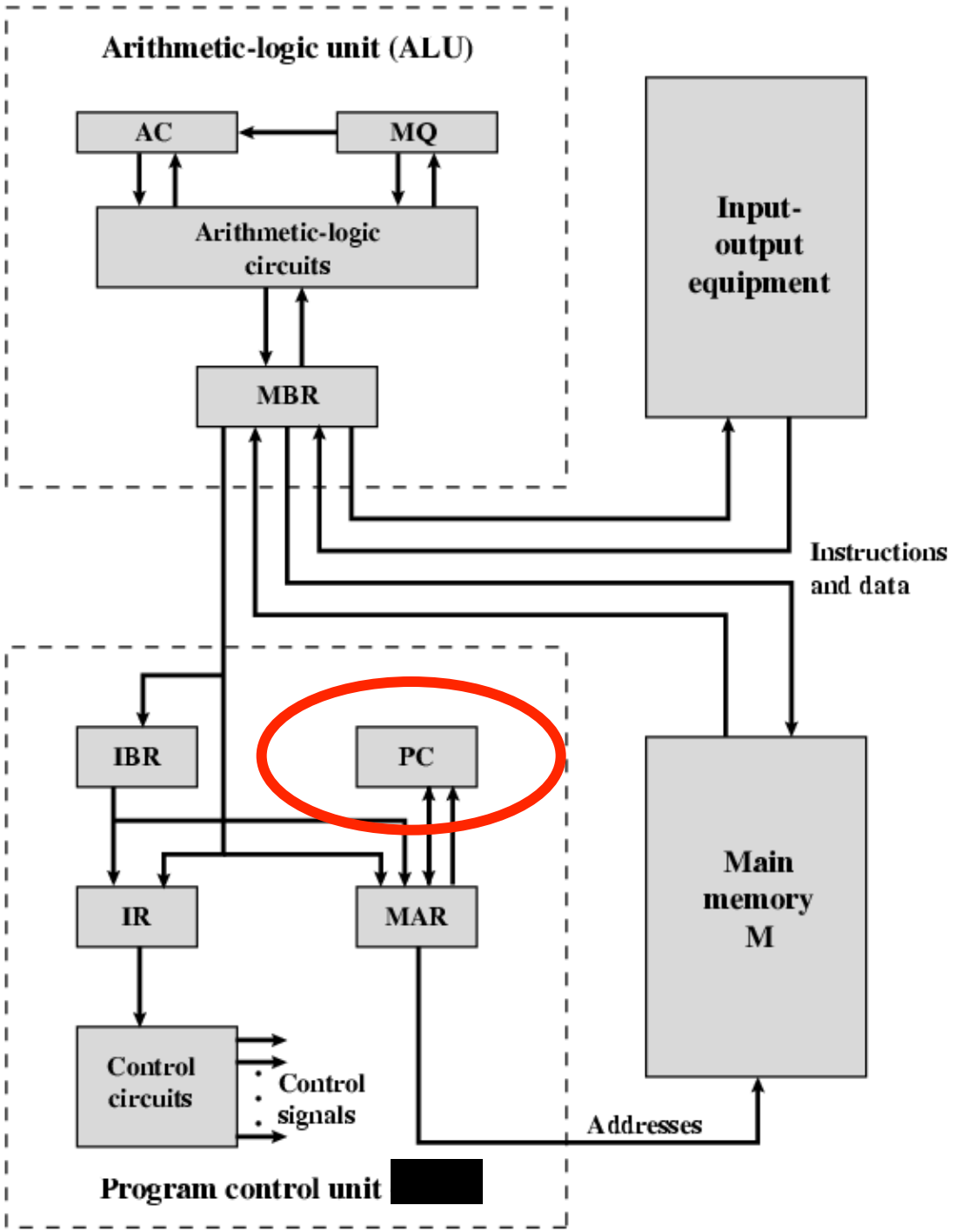
IBR:
Instruction Buffer Register

Empleado para almacenar temporalmente la instrucción contenida en la parte derecha de una palabra en memoria



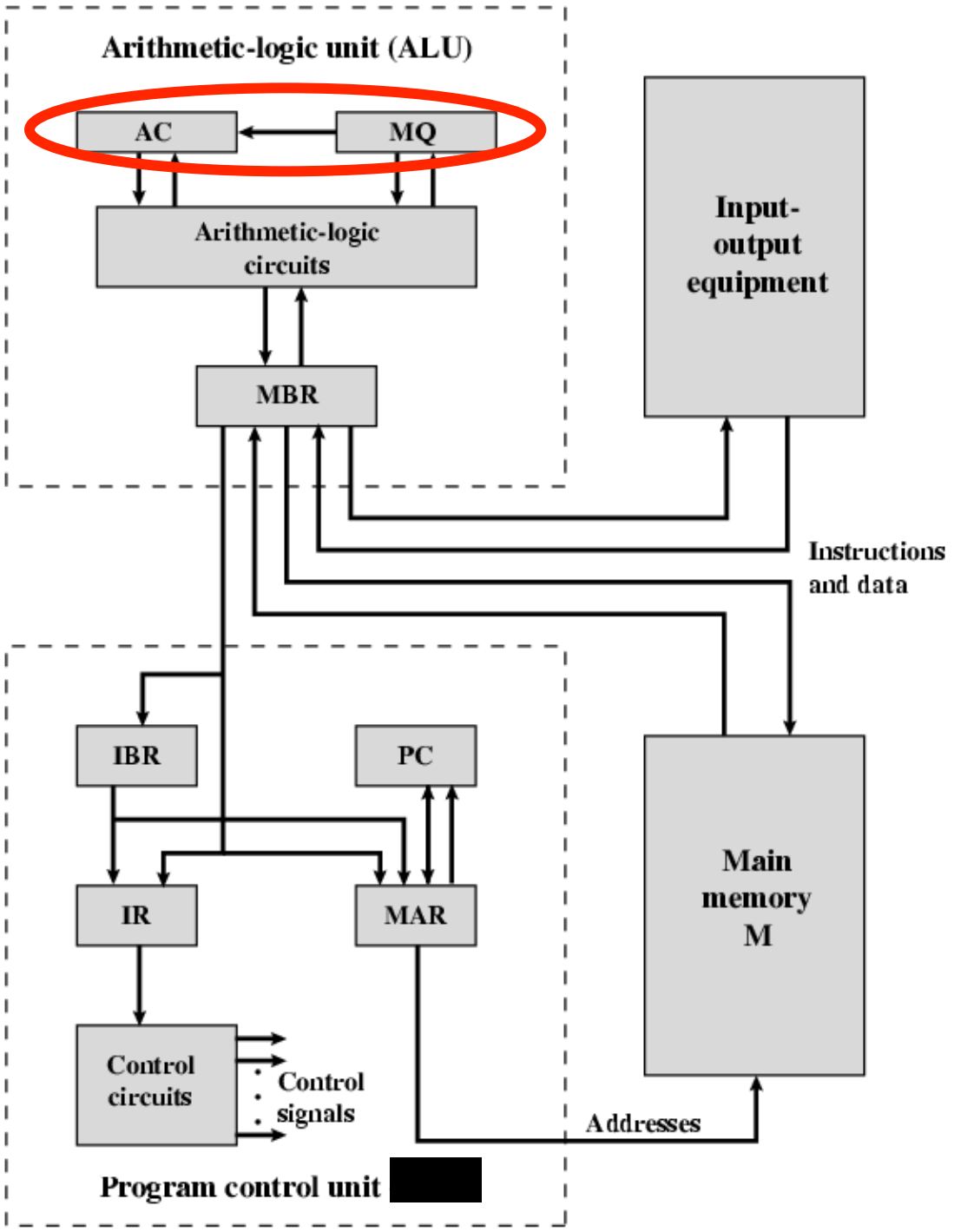
PC:
Program Counter

Contiene la dirección de la próxima pareja de instrucciones que van a ser captadas de la memoria.



AC y MQ:
Accumulator y Multiplier Quotient

Se emplean para almacenar operandos y resultados de operaciones de la ALU temporalmente. Por ejemplo, el resultado de multiplicar dos números de 40 bits es un número de 80 bits; los 40 bits más significativos se almacenan en AC y los menos significativos se almacenan en MQ.



Registros en otras arquitecturas

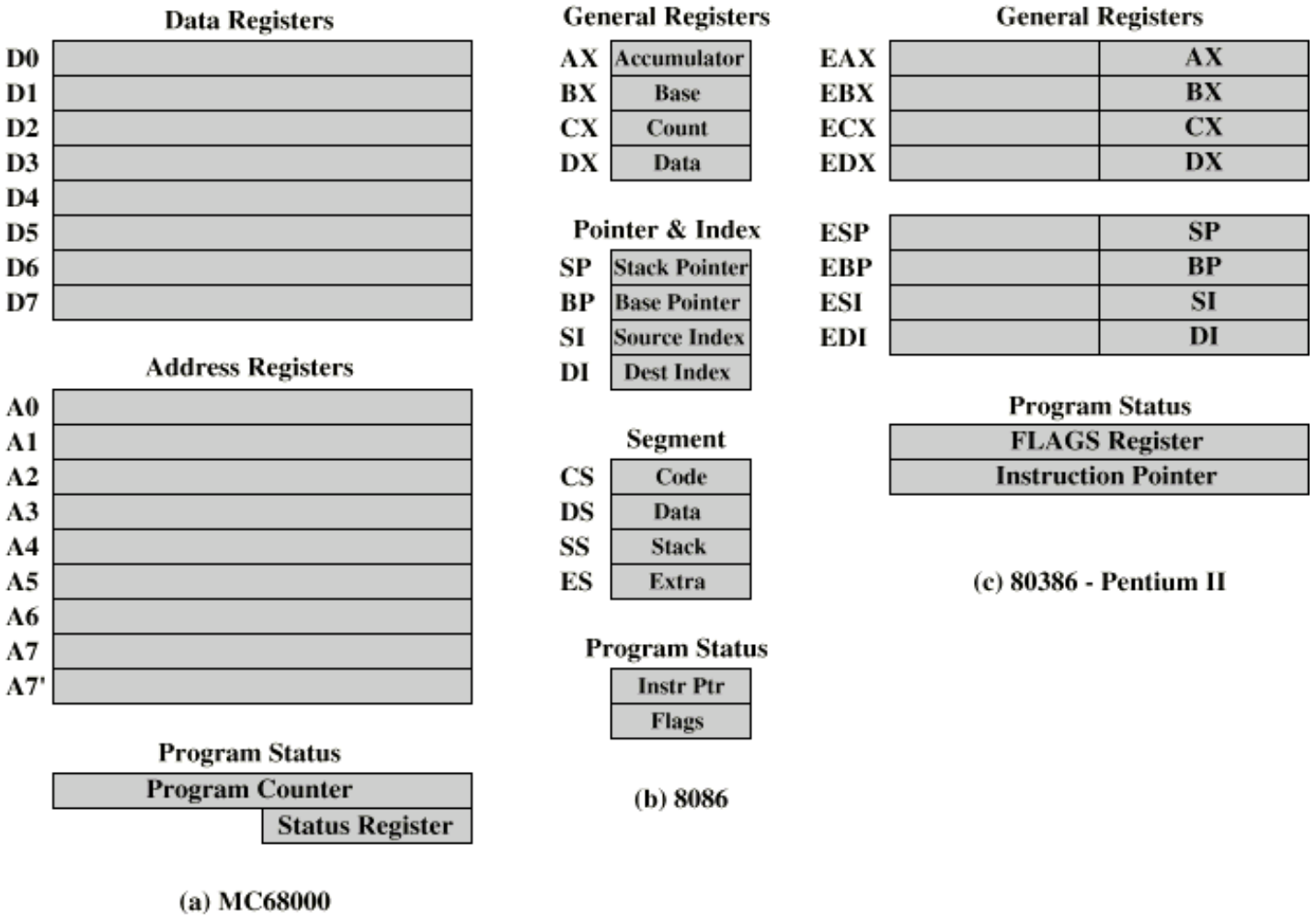
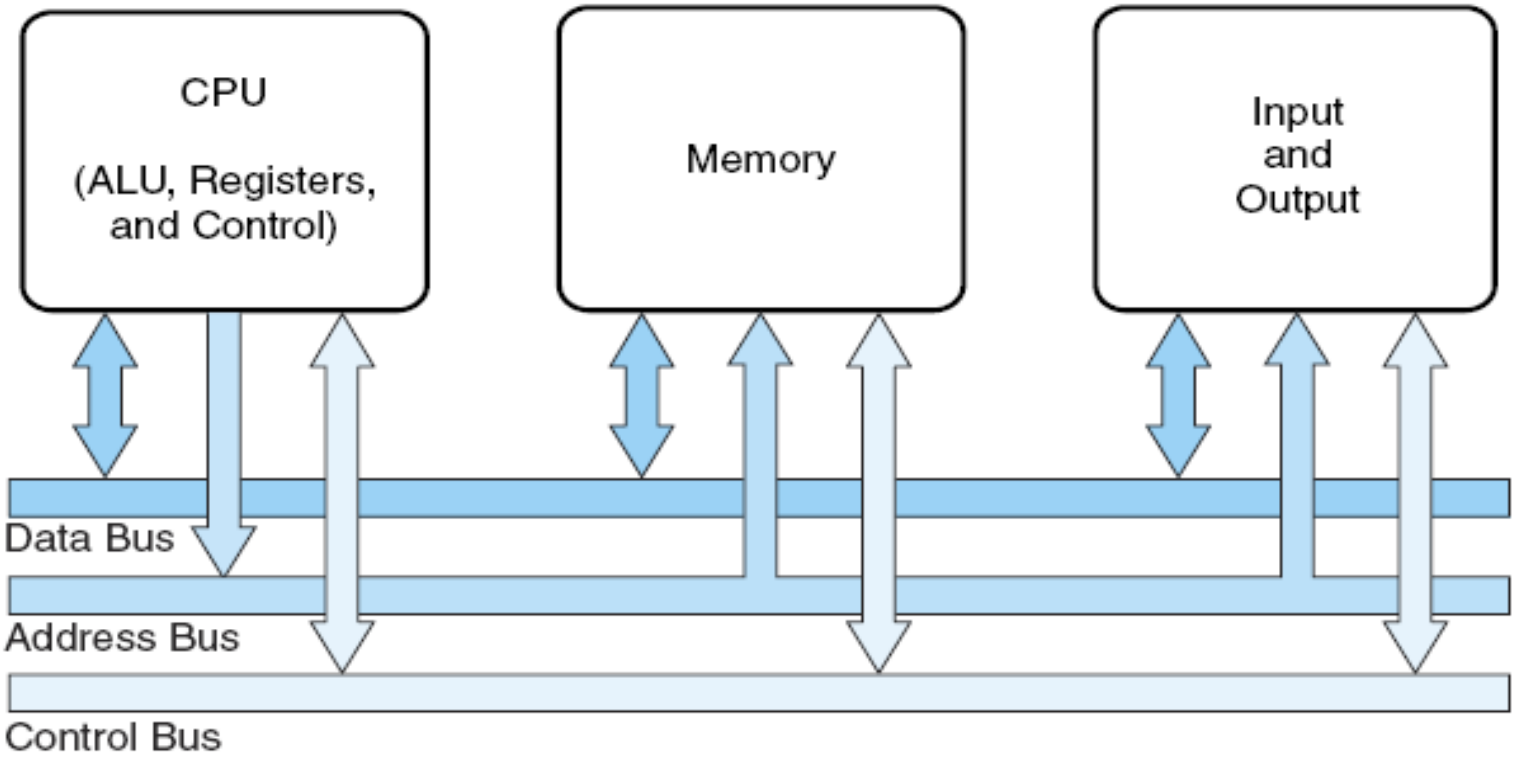
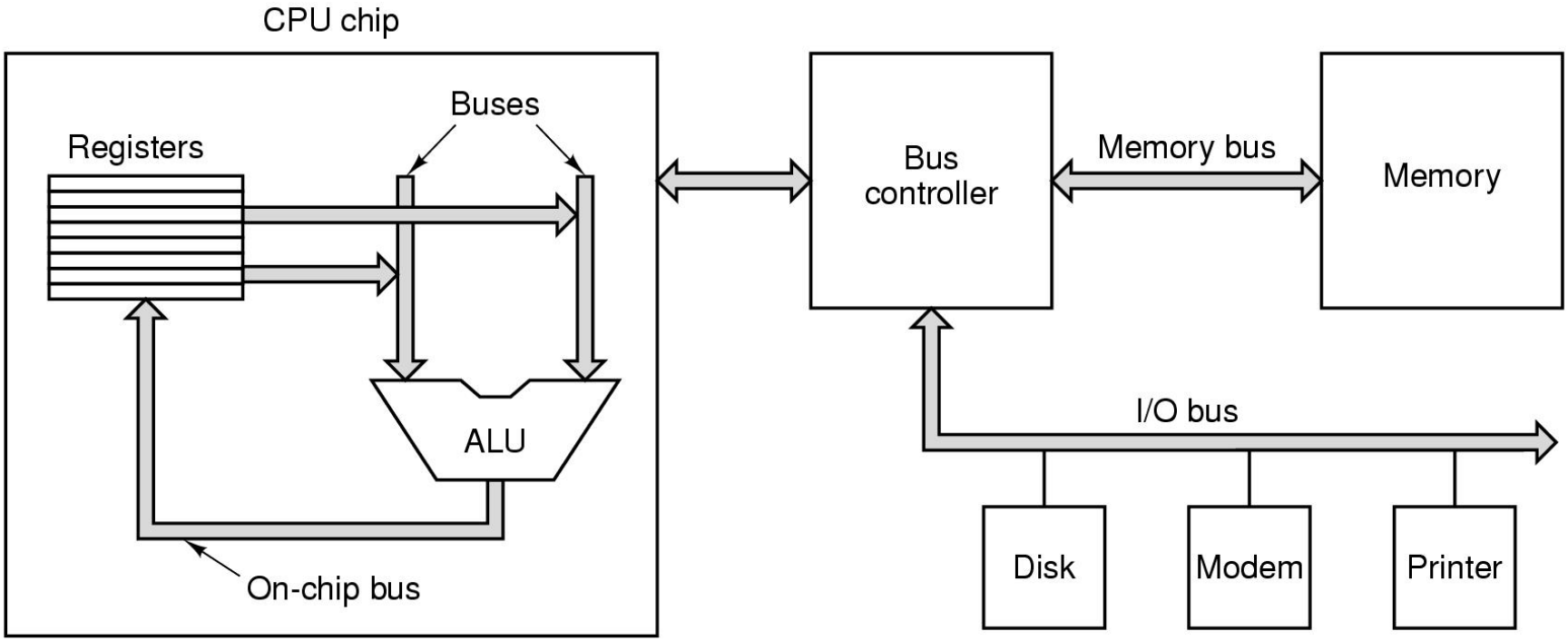


Figure 11.3 Example Microprocessor Register Organizations

Modelo de von Neumann
Bus del Sistema



Una posible configuración



Buses

- Una vía comunicación que conecta 2 o más dispositivos
- En general “broadcast” (todos lo ven)
- En general agrupados
  - Varios canales en un grupo
  - Ej: Data bus de 32 bits, son 32 canales de 1 bit

Data Bus

- Transfieren información
- Su tamaño es un punto clave en la performance del sistema
  - 8, 16, 32, 64 bits

Address bus

- Identifican la fuente o destino de un dato
- Ej: la CPU necesita leer una instrucción (dato) de una locación en memoria
- Su tamaño determina la capacidad máxima de memoria del sistema
  - Ej: el Intel 8080 tiene 16 bit => 64k de espacio direccionable

Control Bus

- Control y sincronización
  - Señal de lectura escritura a memoria
  - Señales del reloj
  - Solicitud de interrupción

Modelos no von Neumann

- Cuello de von Neumann
  - El procesador ejecuta una instrucción por vez…
  - Comunicación con Memoria y E/S empeoran la cosas..
- Mejoras:
  - Buses especializados
  - Interrupciones
  - Unidades de punto flotante
  - Caches,
  - Pipelines
- Otro enfoque: apartarse de la arquitectura clásica de von Neumann.
- Agregar procesadores es una posibilidad…

Modelos no von Neumann

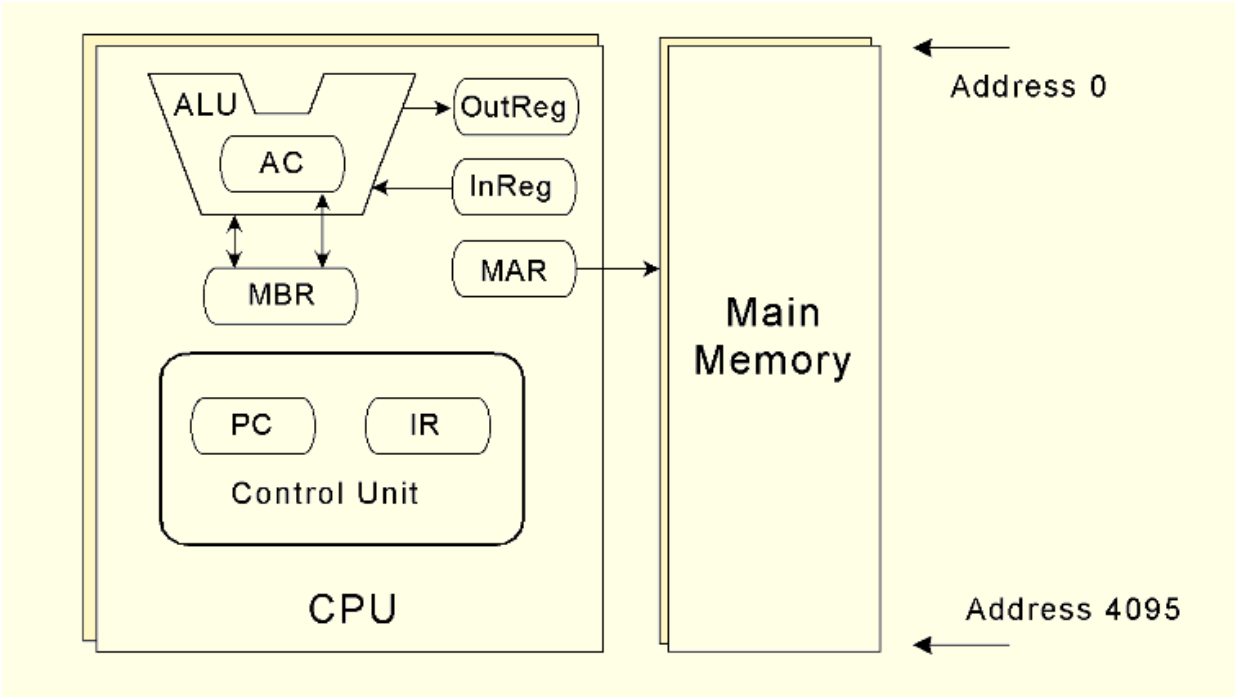
- A finales de los 60’ los sistemas de computo “high-performance” fueron equipados con procesadores duales para mejorar su desempeño.
- En los 70’ supercomputadoras con 32 procesadores.
- En los 80’ con 1000 procesadores
- En 1999, IBM anuncio su sistema “Blue Gene” que contiene aprox. 1 millón de procesadores.

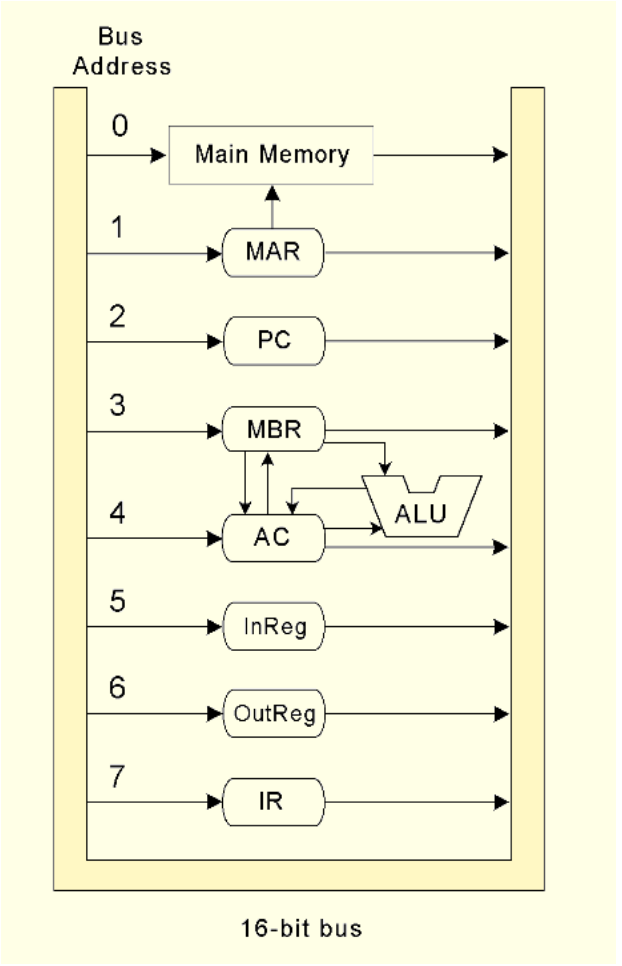
Modelos no von Neumann

- El procesamiento paralelo es una de las formas de mejorar el poder de cómputo.
- Otras ideas más radicales:
  - Computadoras genéticas
  - Computadoras cuánticas
  - Sistemas Dataflow.

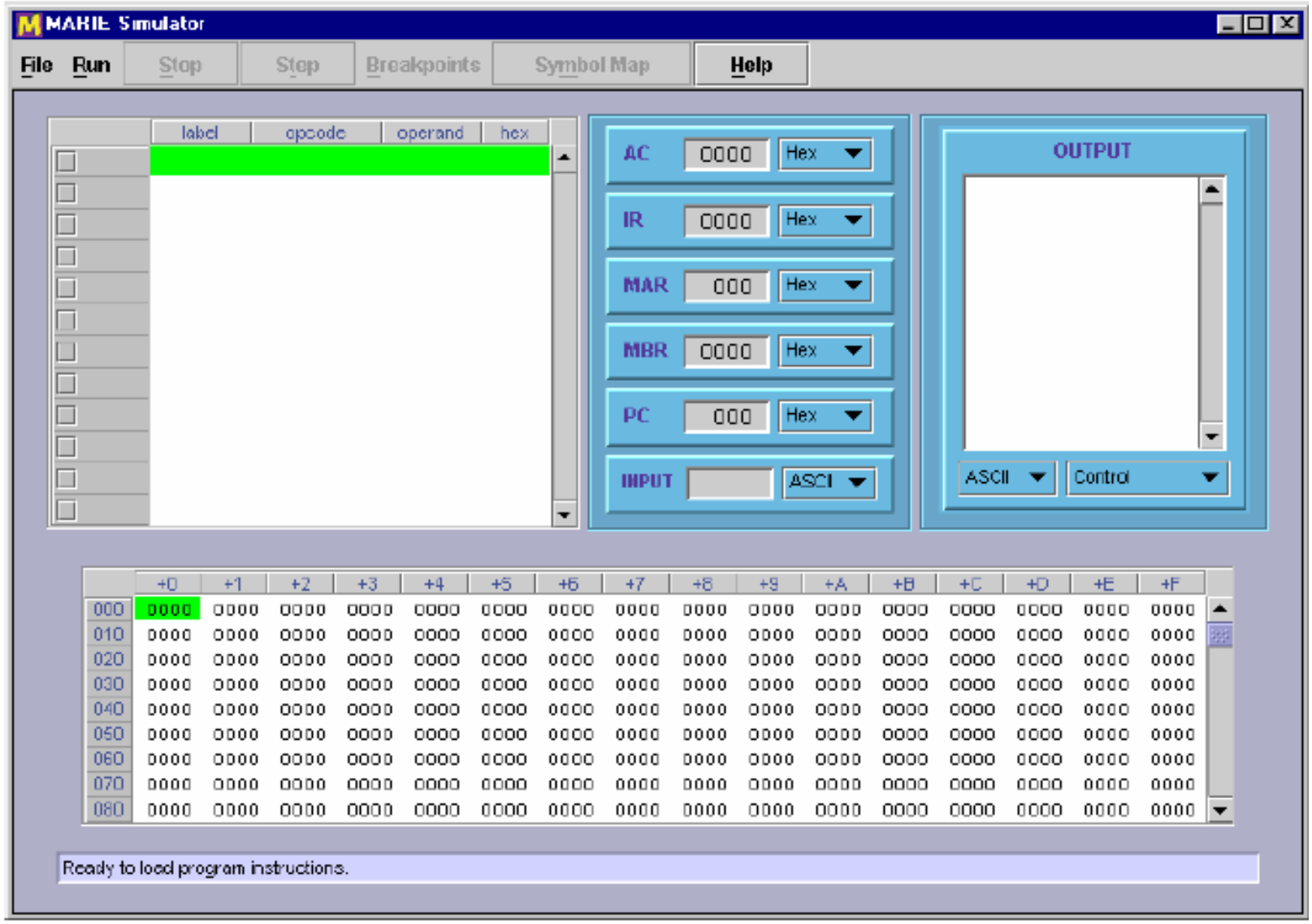
Ejemplo de Arquitectura

- MARIE:





MARIE

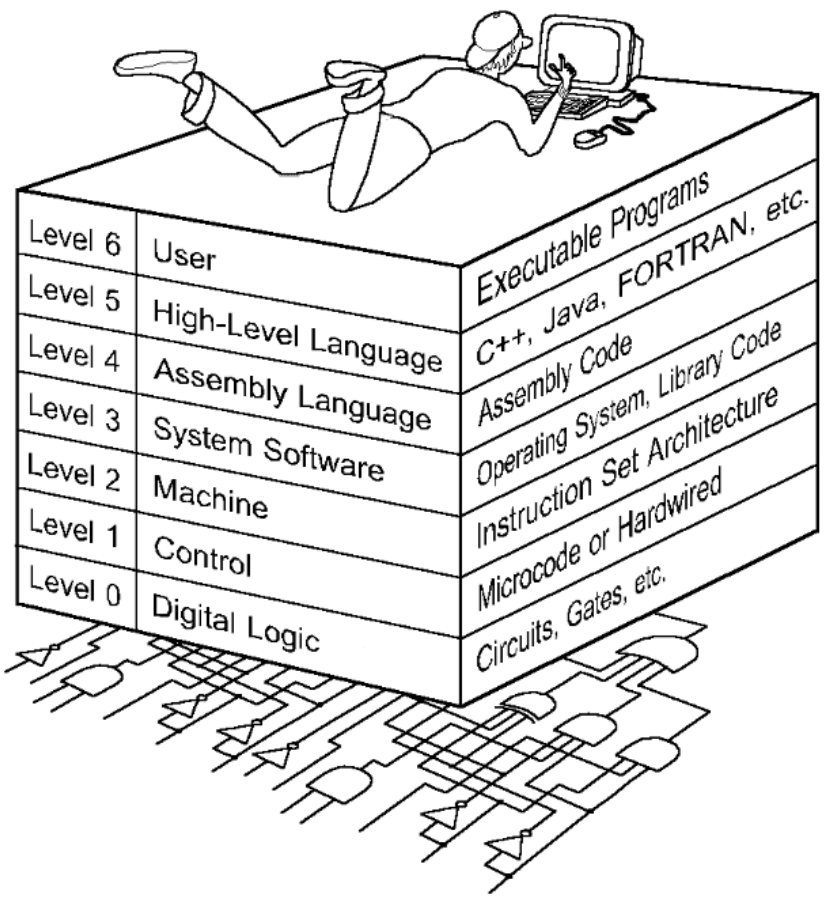

MARIE Simulator
File
Run
Stop
Step
Breakpoints
Symbol Map
Help
label
opcode
operand
hex
AC
0000
Hex
IR
0000
Hex
MAR
000
Hex
MBR
0000
Hex
PC
000
Hex
INPUT
ASCII
OUTPUT
ASCII
Control
+0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F
000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
010 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
020 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
030 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
040 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
050 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
060 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
070 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
080 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
Ready to load program instructions.

La jerarquía de niveles de una computadora

- Una computadora es mucho más que chips.
- Para que la computadora haga “algo” necesita software
- Para escribir programas complejos se suele dividir el problema en módulos que resuelven problemas mas simples
- En las computadoras
  - Varios niveles de máquinas “virtuales”

Jerarquía de niveles

- **Cada capa es una máquina virtual que abstrae a las maquina del nivel inferior.**
- **Las máquinas, en su nivel, "interpretan" sus instrucciones particulares, utilizando servicios de su capa inferior para implementarlas.**
- **En última instancia los circuitos terminan haciendo el trabajo…**



Jerarquía de niveles

- Level 6: Nivel Usuario
  - Ejecución de programas e interfaces de usuario.
  - Pensamos en terminos de la aplicación que se ejecuta
- Level 5: Lenguajes de alto nivel
  - El nivel donde interactuamos cuando escribimos programas en Haskell, C, Java, etc.
  - Pensamos el algoritmos, TADs, etc.

Jerarquía de niveles

- Level 4: Nivel de Lenguaje Ensamblador
  - Lenguaje ensamblador, en general producido por compiladores, o escrito directamente por programadores.
  - Muy cercano a la arquitectura de la computadora.
- Level 3: Nivel del software del Sistema
  - Controla la ejecución de los procesos del sistema.
  - Protege los recursos.
  - Brinda servicios para acceder a dispositivos de E/S
  - Muchas instrucciones en Assembler pasan este nivel sin modificación.

Jerarquía de niveles

- Level 2: Nivel del Lenguaje de máquina
  - También conocido como nivel ISA (Instruction Set Architecture).
  - Consiste en las instrucciones particulares para la arquitectura de la maquina.
  - Los programas escritos en lenguaje de maquina no necesitan compilación ni ensamblado.

Jerarquía de niveles

- Level 1: Nivel de Control
  - La unidad de control (UC) decodifica y ejecuta instrucciones y mueve datos a traves del sistema.
  - Puede ser microprogramada o “cableada”.
    - Un microprograma es un programa escrito en un lenguaje de bajo nivel que puede ser implementado enl hardware.
    - Las UC “cableadas” tienen hardware que ejecuta directamente las instrucciones en codigo de maquina

Hardwired vs. Micro-programada

Hardwired	Micro-programada
☺ Muy rápida, es un flujo directo	☹ Interpretar instrucciones toma tiempo
☹ Redes muy complejas de implementar	☺ Programación estándar, escalable
☹ No puede modificarse	☺ Es posible hacer upgrade del programa
☹ Amarrado a la arquitectura	☺ Flexible, varias implementaciones

Jerarquía de niveles

- Level 0: Nivel de Lógica Digital
  - Aquí encontramos los circuitos digitales (chips).
  - Son básicamente compuertas y cables.
  - Implementan la lógica matemática de los niveles superiores.

Links

- http://www.turing.org.uk
- John von Neumann, “First Draft of a Report on the EDVAC”, 1946 (en sección download)
- Computer Architecture home page: www.cs.wisc.edu/~arch/www
- Null, L. and J. Lobur. The Essentials of Computer Organization and Architecture, Jones and Bartlett Publishers, Feb. 2003