

Tema 1 Anexo – Arquitectura de Computadoras

Objetivo de la presentación

- Incluye los conceptos vistos en asignatura/s previa/s y que son necesarios tener presentes para conceptos que se verán durante la cursada.

Elementos Básicos de una computadora

- Procesador
- Memoria Principal
 - Volátil
 - Se refiere como memoria real o primaria
- Componentes de E/S
 - Dispositivos de memoria secundaria
 - Equipamiento de comunicación
 - Monitor / teclado / mouse
- Bus Sistema
 - comunicación entre procesadores, memoria, dispositivos de E/S

Componentes de alto nivel

1. Bloques principales

- **CPU (Unidad Central de Procesamiento)**

Es el “cerebro” del sistema, donde se ejecutan las instrucciones. Dentro de la CPU aparecen registros clave:

- **PC (Program Counter / Contador de programa):** guarda la dirección de la **siguiente instrucción** a ejecutar.
- **IR (Instruction Register / Registro de instrucción):** contiene la **instrucción actual** que se está ejecutando.
- **MAR (Memory Address Register / Registro de direcciones de memoria):** almacena la **dirección en memoria** a la que la CPU quiere acceder.
- **MBR (Memory Buffer Register / Registro intermedio de memoria):** guarda los **datos/instrucciones** que vienen de memoria o que van hacia ella.
- **I/O AR e I/O BR:** cumplen un papel similar a MAR y MBR, pero enfocados en operaciones de **entrada/salida**.

👉 La CPU también tiene la **Unidad de ejecución**, donde realmente se procesan los cálculos y operaciones lógicas.

- **Main Memory (Memoria Principal / RAM)**

Es donde se guardan tanto:

- **Instrucciones del programa** (qué hacer).
- **Datos** (sobre qué hacerlo).

Cada celda de memoria está numerada con direcciones (0, 1, 2, ...). La CPU accede a memoria usando el **MAR** para decir “quiero la posición X” y recibe el contenido en el **MBR**.

- **I/O Module (Módulo de entrada/salida)**

Aquí se gestionan los periféricos (teclado, disco, pantalla, etc.).

Usa **buffers** para “amortiguar” la transferencia de datos entre los dispositivos lentos y la memoria/CPU más rápida.

Por eso hay registros de entrada/salida:

- **I/O AR (Input/Output Address Register):** dirección del dispositivo al que quiero acceder.
 - **I/O BR (Input/Output Buffer Register):** datos que van o vienen del dispositivo.
-

- **System Bus (Bus del sistema)**

Es como la **autopista** que conecta CPU, memoria e I/O. Se suele dividir en:

- **Bus de datos:** por donde viajan datos e instrucciones.
 - **Bus de direcciones:** por donde se indica “a qué dirección de memoria o dispositivo acceder”.
 - **Bus de control:** señales de lectura, escritura, interrupción, etc.
-

2. Flujo típico

1. El **PC** apunta a la dirección de la próxima instrucción.
 2. Esa dirección pasa al **MAR** y se lee de memoria.
 3. El contenido llega al **MBR** y de ahí al **IR**.
 4. La **Unidad de ejecución** interpreta la instrucción y, si necesita datos, los busca en memoria o en un dispositivo de I/O.
 5. Los datos viajan por el bus y se procesan.
 6. Finalmente, el resultado puede guardarse de nuevo en memoria o enviarse a un dispositivo.
-

3. Idea clave

La figura muestra que un computador es un **sistema organizado de componentes que se comunican a través del bus**. La CPU coordina todo, pero depende de la memoria para instrucciones/datos y de los módulos de E/S para interactuar con el exterior.

En resumen:

Esta imagen es una **arquitectura básica de von Neumann**, donde programas e información se almacenan en la misma memoria y la CPU los ejecuta paso a paso.

Registros del Procesador

- Visibles por el usuario
 - Registros que pueden ser usados por las aplicaciones
- De Control y estado
 - Para control operativo del procesador
 - Usados por rutinas privilegiadas del SO para controlar la ejecución de proces

Registros Visibles por el usuario

- Pueden ser referenciados por lenguaje de máquina
- Disponible para programas/aplicaciones
- Tipos de registros
 - Datos
 - Direcciones
 - Index
 - Segment pointer
 - Stack pointer

Registros de Control y Estado

- Program Counter (PC)
 - Contiene la dirección de la proxima instrucción a ser ejecutad
- Instruction Register (IR)
 - Contiene la instrucción a ser ejecutada
- Program Status Word (PSW)
 - Contiene códigos de resultado de operaciones
 - habilita/deshabilita Interrupciones
 - Indica el modo de ejecución (Supervisor/usuario)

Ciclo Ejecución de Instrucción

- Dos pasos
 - Procesador lee la instrucción desde la memoria

- Procesador ejecuta la instrucción

Ciclo Instrucción

Instrucción: Fetch y Execute

- El procesador busca (fetch) la instrucción en la memoria
 - $(PC) \rightarrow IR$
- El PC se incrementa después de cada fetch para apuntar a la próxima instrucción
 - $PC = PC + 4$

IR - Instruction Register

- La instrucción referenciada por el PC se almacena en el IR y se ejecuta
- Categorías de instrucciones
 - Procesador - Memoria
 - Transfiere datos entre procesador y memoria
 - Procesador - E/S
 - Transfiere datos a/o desde periféricos
 - Procesamiento de Datos
 - Operaciones aritméticas o lógicas sobre datos
 - Control
 - Alterar secuencia de ejecución

Interrupciones

- Interrumpen el secuenciamiento del procesador durante la ejecución de un proceso
- Dispositivos de E/S más lentos que el procesador
 - Procesador debe esperar al dispositivo

Necesidades del SO

- Postergar el manejo de una interrupción en momentos críticos
- Atender en forma eficiente: la rutina de atención adecuada según el dispositivo
- Tener varios niveles de interrupción para que el SO pueda distinguir entre interrupciones de alta prioridad y de baja prioridad y responder adecuadamente

Clases de Interrupciones

1. Program (del programa)

- Ocurren **dentro de la CPU** al ejecutar una instrucción.
- Ejemplos:
 - Overflow aritmético (cuando el resultado es demasiado grande).
 - División por cero.
 - Instrucción ilegal (código no válido).
 - Acceso a memoria no permitido.

👉 Suelen indicar **errores de software** o mal uso del programa.

1. **Timer (temporizador interno)**

- Generadas por un **reloj del procesador**.
 - Sirven para que el SO ejecute tareas periódicas.
 - Ejemplo clásico: el **time-sharing** (repartir CPU entre procesos).
- 👉 Son controladas por el SO, no por el programa del usuario.
-

1. **I/O (entrada/salida)**

- Generadas por un **controlador de dispositivo**.
 - Usadas para:
 - Avisar que una operación terminó (ej: “la impresora ya imprimió”).
 - Reportar un error (ej: “disco no encontrado”).
- 👉 Permiten que la CPU no se quede esperando: sigue trabajando y se “entera” por la interrupción cuando el dispositivo terminó.
-

1. **Hardware failure (fallo de hardware)**

- Son las más críticas: indican un **fallo físico**.
 - Ejemplos:
 - Corte de energía.
 - Error de memoria (paridad, bits dañados).
- 👉 El SO suele reaccionar entrando en “modo seguro” o reiniciando, porque puede comprometer la estabilidad del sistema.
-

Interrupt Handler

- Programa (o rutina) que determina la naturaleza de una interrupción y realiza lo necesario para atenderla
 - Por ejemplo, para un dispositivo particular de E/S
- Generalmente es parte del SO

Interrupciones

- Suspende la secuencia normal de ejecución

Ciclo de interrupción

- El procesador chequea la existencia de interrupciones.
- Si no existen interrupciones, la próxima instrucción del programa es ejecutada
- Si hay pendiente alguna interrupción, se suspende la ejecución del programa actual y se ejecuta la rutina de manejo de interrupción

Procesamiento simple de una interrupción

- Flujo en **Hardware**

1. **Device controller u otro hardware genera una interrupción**

Ejemplo: el disco terminó de leer datos, o el temporizador venció.

2. **El procesador termina la instrucción actual**

Muy importante: la CPU nunca interrumpe a mitad de una instrucción → primero la finaliza.

3. **El procesador envía señal de reconocimiento (acknowledge)**

Para avisar al dispositivo: “Recibí tu interrupción, ahora me ocupo”.

4. **El procesador guarda el estado mínimo (PSW y PC en la pila)**

- **PC (Program Counter):** dónde estaba el programa antes de la interrupción.
- **PSW (Program Status Word):** contiene flags y modo de ejecución.

👉 Esto es **fundamental**: permite **retomar luego la ejecución original** sin perder nada.

5. **El procesador carga el nuevo PC con la dirección del manejador de interrupción**

Es decir, salta al código del **interrupt handler** (rutina de atención a interrupciones).

- Flujo en **Software**

1. **Se guarda el resto del estado del proceso**

Los registros y variables que el proceso estaba usando.

👉 Así, el SO puede restaurar todo luego.

2. **Se procesa la interrupción**

El código del manejador hace lo que corresponde: leer datos de un dispositivo, actualizar un contador de tiempo, manejar un error, etc.

3. **Se restaura el estado del proceso**

Se cargan nuevamente los registros guardados.

4. **Se restaura el viejo PSW y PC**

El procesador recupera el punto exacto donde había quedado el programa interrumpido → y lo continúa como si nada hubiera pasado.

- Idea central

Este ciclo asegura dos cosas:

1. La **CPU atiende el evento urgente**.
2. Luego **retoma exactamente** la ejecución normal del programa, transparente para el usuario.

Interrupciones no enmascarables y enmascarables

- La mayoría de las CPUs tienen dos líneas de requerimiento de interrupciones: la de no enmascarables y las enmascarables.
- La de no enmascarables se reservan para eventos tales como errores de memoria no recuperables.
- La de enmascarables puede ser “apagada” por la CPU si en ese momento se está ejecutando una secuencia crítica de instrucciones. Estas son las que usan los controladores de dispositivo cuando necesitan servicio

Múltiples Interrupciones

- Deshabilitar las interrupciones mientras una interrupción está siendo procesada
- Definir prioridades a las interrupciones

Descripción de número en el vector de interrupciones de procesador Intel:

- De 1 a 31: no enmascarables, por ejemplo, errores de condición
- 32 a 255: enmascarables, usadas para interrupciones generadas por dispositivos