Sistemas y Aplicaciones Informáticas

Tema 03. Componentes, Estructura y Funcionamiento de la Unidad Central de Proceso.

1. Ámbito de docencia.

- Sistemas informáticos monousuario y multiusuario (ASI 1).
- Sistemas informáticos multiusuario y en red (DAI 1).
- Instalación y mantenimiento de equipos y sistemas informáticos (ESI 1).

2. Unidad central de proceso (CPU).

2.1. Arquitectura de Von Neumann.

- La arquitectura de un ordenador es el conjunto de funcionalidades y características que ofrece al usuario, mientras que su estructura está formada por los componentes lógicos que llevan a cabo dichas funcionalidades. La tecnología de un ordenador es una versión concreta de la estructura que determina los componentes físicos que se van a utilizar y cómo se van a interconectar.
- La arquitectura de Von Neumann se caracteriza por lo siguiente:
  - El ordenador dispone de una memoria principal en la que se almacenan simultáneamente instrucciones y datos sin una distinción explícita. Se puede acceder a la información contenida en la memoria especificando la dirección donde se encuentra almacenada.
  - Las instrucciones están formadas por un código binario que indica una operación determinada, y los datos están representados por los operandos de la instrucción. El código de la instrucción define la cantidad y el modo de acceso a los operandos.
  - Las instrucciones se ejecutan una tras otra según su posición en la memoria, aunque existe la posibilidad de romper el flujo secuencial mediante instrucciones de salto. Para ello se dispone de un registro que almacena la dirección de la siguiente instrucción a ajecutar.

2.2. Descripción de la CPU. Características.

- La unidad central de proceso (CPU) es el elemento funcional de la arquitectura de Von Neumann que se encarga del control y de la ejecución de cada una de las operaciones que se efectúan dentro del ordenador con el fin de realizar el tratamiento automático de la información.
- Las características que permiten diferenciar una CPU de otra son las siguientes:
  - *Velocidad de proceso*. La CPU recibe una señal de un oscilador que sincroniza todas las operaciones que se realizan. Cuanto mayor sea la frecuencia del oscilador, más rápido será el procesamiento de las instrucciones. La velocidad de proceso se mide en Hz.
  - *Juego de instrucciones*. Está formado por el conjunto de códigos de instrucción que definen las operaciones que puede realizar la CPU. El código máquina generado a partir de un determinado programa debe adaptarse al juego de instrucciones particular de cada CPU.
  - *Ancho del bus de datos*. Se mide en bits, y determina la palabra o la cantidad máxima de memoria a la que puede acceder la CPU en una sola operación de lectura o escritura.
  - *Ancho del bus de direcciones*. Se mide en bits, y determina la cantidad máxima de posiciones de memoria que puede direccionar la CPU.
  - *Número de registros internos*. La CPU utiliza un conjunto de registros para almacenar datos e instrucciones temporalmente. Su número está en función del juego de instrucciones.
  - *Número de líneas de interrupción*. Indica la capacidad que tiene la CPU para responder ante operaciones de entrada/salida (interrupciones E/S) o eventos imprevistos (excepciones).

3. Componentes y estructura de la CPU.

3.1. Unidad de control (CU).

3.1.1. Funciones.

- Se encarga del gobierno y funcionamiento del ordenador, gestionando la comunicación entre los componentes del ordenador y resolviendo las situaciones anómalas que puedan ocurrir.
- Sus funciones más importantes son las siguientes:
  - Obtener de la memoria las instrucciones e interpretarlas.
  - Obtener de la memoria los datos correpondientes a cada instrucción.
  - Generar y controlar la secuencia de acciones de cada instrucción.
  - Controlar el flujo de datos entre las diferentes partes que forman el ordenador.
  - Enviar a la memoria los resultados de las operaciones realizadas.
  - Enviar y recibir señales de control de periféricos externos.

3.1.2. Componentes.

- **Registro de instrucciones**. Contiene la instrucción que se está ejecutando. Consta del código de la operación que se va a realizar, el modo de direccionamiento de la memoria para acceder a la información que se va a procesar y el campo de dirección efectiva de la información.
- **Registro contador de programa**. Contiene la dirección de memoria de la siguiente instrucción a ejecutar. Normalmente este contador se incrementa en cuanto la memoria principal acepta la dirección anterior, aunque pueden existir instrucciones de salto de secuencia.
- **Registro de direcciones de memoria**. Se utiliza para almacenar direcciones de memoria en las que se puede leer los datos de una instrucción o escribir el resultado de una operación.
- **Registro de estado**. Contiene información sobre el resultado de la operación anterior y de posibles situaciones anómalas o especiales, tales como desbordamiento, interrupciones, errores de paridad, etc., que exigen una acción inmediata por parte de la unidad de control.
- **Decodificador de instrucciones**. Interpreta el contenido del registro de instrucciones y genera el cronograma de señales de control necesarias para ejecutar la instrucción.
- **Decodificador de direcciones**. Obtiene la posición de memoria a la que hay que acceder a partir del código de la instrucción a ejecutar y de sus operandos.
- **Secuenciador**. Se encarga de distribuir entre los elementos del sistema el cronograma de señales de control necesarias para ejecutar la instrucción.
- **Reloj**. Es un circuito oscilador que genera automáticamente una señal en forma de pulsos. La señal del reloj representa la temporización básica del sistema, puesto que determinan el menor tiempo que puede durar una operación elemental. Se llama ciclo de reloj a la duración de un tiempo elemental determinado entre dos pulsos consecutivos de reloj.

3.1.3. Implementación de la CU. Tipos de operaciones.

- El diseño de la CU exige la definición de las señales que hay que activar en cada una de las instrucciones. Se trata por tanto de un circuito combinacional que convierte el código de instrucción en señales de control que varían sincronizadamente con la señal del reloj de la CU.
- Existen dos maneras de implementar la creación de las señales de control:

    - *Mediante lógica almacenada*. Consiste en emplear una memoria que almacenar el estado de las señales de control en cada instante de la ejecución de cada instrucción, lo que se denomina firmware. Por tanto, para generar el cronograma de señales de control basta con ir leyendo las sucesivas palabras de dicha memoria, lo que permite mayor flexibilidad.
    - *Mediante lógica cableada*. La CU se construye con puertas lógicas y las señales de control se generan a partir de las mismas. Su diseño es muy laborioso y su modificación exige un rediseño completo. Sin embargo, es mucho más rápida que las CU de lógica almacenada.
- Los tipos de operaciones que realiza la CU son las siguientes:
    - *Transferencia*. Requieren un registro origen y otro destino. En estas operaciones se establece el camino físico entre el origen y el destino, y se envían las señales adecuadas para que el destino cargue la información sin modificar el origen.
    - *Proceso*. Requieren uno o dos registros origen y otro destino. Son análogas a las anteriores, excepto en que la información que llega al destino es producto de una determinada operación realizada con la información de origen.

3.2. Unidad aritmético-lógica (ALU).

3.2.1. Funciones. Operaciones.

- Es la unidad funcional encargada de realizar las operaciones aritméticas y lógicas bajo la supervisión de la unidad de control. Las operaciones que puede efectuar esta unidad son muy elementales, puesto que normalmente está formada por un circuito sumador-restador. Por tanto, la ejecución de operaciones complejas se lleva a cabo descomponiéndolas en pasos elementales.
- Las operaciones que es capaz de realizar la ALU se clasifican en tres grupos:
    - *Aritméticas*. Normalmente son la suma (ADD), la resta (SUB), la multiplicación (MUL), la división (DIV) y el cambio de signo.
    - *Lógicas*. Normalmente son la negación (NOT), la suma lógica (OR), el producto lógico (AND) y la suma exclusiva (XOR).
    - *Desplazamiento*. Consiste en desplazar los bits de una palabra un determinado número de posiciones hacia la derecha o hacia la izquierda. El desplazamiento puede ser:
        - **Lógico**. En cada desplazamiento se pierde un bit por un extremo, reemplazado por 0 ó 1 en el extremo opuesto. Este tipo de desplazamiento implica pérdida de información.
        - **Aritmético**. Actúa igual que el desplazamiento lógico, pero en este caso el bit de signo no se desplaza y se mantiene su valor.
        - **Circular**. Los bits que se introducen por un extremo son los mismos y en el mismo orden que los que van desapareciendo por el extremo opuesto. Este tipo de desplazamiento no implica pérdida de información.

3.2.2. Componentes.

- **Registros de operandos**. Son aquellos que almacenan los operandos de la operación.
- **Acumulador**. Es un registro especial donde se almacena el resultado de las operaciones.
- **Circuitos operacionales**. Son los circuitos capaces de realizar las operaciones de la ALU. Normalmente existen varios circuitos generales para realizar todo tipo de operaciones y un circuito especializado para realizar las operaciones de coma flotante. Se componen de:

    - *Circuitos sumadores*. Encargados de realizar cualquier operación aritmética, puesto que todas las operaciones se convierten en sumas. Éstas se realizan en serie utilizando un solo circuito para todos los dígitos, o en paralelo utilizando un circuito para cada dígito.
    - *Circuitos complementadores*. Son necesarios para realizar las restas de números. Para ello estos circuitos calculan el complemento a uno o a dos del sustraendo, y el resultado se suma al minuendo mediante un circuito sumador.
    - *Circuitos lógicos*. Realizan operaciones lógicas (AND, OR…) y de comparación (igual, mayor…) entre dos operandos bit a bit produciendo un resultado booleano.
- **Registro de estado**. Almacena un conjunto de flags relativos a la última operación realizada:
    - *Cero*. Este indicador se pone a uno cuando el resultado ha sido cero.
    - *Negativo*. Si este bit es uno el resultado de la operación es negativo.
    - *Acarreo*. Si el resultado tiene acarreo aritmético, el indicador se pone a uno.
    - *Desbordamiento*: Se pone a uno si el resultado produce desbordamiento.
    - *Paridad*. En el caso de emplear la paridad como criterio para comprobar si un dato es o no correcto, este indicador realiza la verificación de forma inmediata, poniéndose a uno o a cero dependiendo de que se cumpla o no el criterio de paridad establecido.

3.3. Registros internos. Tipos.

- Son pequeñas memorias dedicadas al almacenamiento temporal de datos. Normalmente su tamaño es similar al ancho del bus de datos, aunque a veces son de menor tamaño. Dentro de la jerarquía de las memorias, los registros son la memoria a la que se accede con mayor rapidez.
- Existen dos tipos de registros:
    - *Registros de uso general*. Almacenan cualquier tipo de información. Se dividen en:
        - **Registros de datos**. Almacenan los datos que son utilizados frecuentemente o los resultados intermedios de operaciones. Por ejemplo, el acumulador de la ALU.
        - **Registros de direcciones**. Almacenan la dirección de memoria donde se encuentra un dato o donde se desea guardar. Contienen un valor índice por el cual se puede acceder a posiciones contiguas de memoria, únicamente incrementando el valor del índice.
    - *Registros específicos*. Son aquellos que tienen una función específica, como el contador de programa, el registro de instrucción o el registro de estado. Un registro específico especial es el puntero de pila, que Contiene la dirección de memoria del último dato almanenado en la pila, por lo que su valor aumenta o disminuye según el número de elementos que contenga. Si la pila está vacía, contiene la dirección de comienzo o base de la pila. Otro puntero contiene siempre la dirección base de la pila.

4. Funcionamiento de la CPU.

4.1. Ejecución de una instrucción.

- La ejecución de una instrucción se desarrolla en una secuencia de operaciones más elementales, como la escritura de un dato en memoria o una operación lógica en la ALU. Cada una de estas operaciones elementales requiere la activación de las correspondientes señales de control procedentes del secuenciador de la unidad de control, de manera sincronizada con el reloj.

- Normalmente la ejecución de una instrucción se desarrolla en dos fases diferenciadas:
  - *Fase de búsqueda*. Comprende el conjunto de operaciones elementales que han de realizarse para llevar la instrucción a ejecutar desde la posición de memoria en que se encuentre hasta el registro de instrucción de la unidad de control. Se desglosa en los siguientes pasos:
    - **Obtención de la dirección de la instrucción**. Transferencia del contenido del contador de programa al registro de direcciones de memoria.
    - **Obtención de la instrucción**. Ejecución de un ciclo de lectura de la memoria y transferencia del dato leído al registro de instrucción.
    - **Incremento del contador del programa**. Para apuntar a la siguiente instrucción a ejecutar o al primer operando de la instrucción actual.
  - *Fase de ejecución*. Comprende el conjunto de operaciones elementales específicas de la instrucción a ejecutar. Dependiendo de la complejidad de la instrucción, esta fase podrá descomponerse en varias subfases. En general se desglosa en los siguientes pasos:
    - **Decodificación de la instrucción**. Análisis de la instrucción para determinar el tipo de operación que se va a efectuar y los operandos que se necesitan.
    - **Obtención de los operandos**. A partir del código de la instrucción se determina si hay que obtener los operandos a través de su dirección de manera similar a la obtención del código de la instrucción, o están incluidos en la instrucción.
    - **Ejecución de la operación**. Se realiza la operación indicada en los circuitos de la ALU.
    - **Almacenamiento del resultado**. El contenido del acumulador de la ALU se envía a la dirección de memoria o al registro indicado en uno de los operandos de la instrucción.

4.2. Tratamiento de interrupciones.

- Una interrupción es un mecanismo con el que se puede detener temporalmente el flujo normal del programa en ejecución, al que debe responder el sistema en un tiempo finito y especificado. Cuando se produce una interrupción, el flujo normal de procesamiento es modificado por un suceso que necesita un servicio inmediato.
- Después de cada instrucción la CPU verifica la línea de interrupción. Si se encuentra activa indica que se ha producido una interrupción, en caso contrario se pasa a la siguiente instrucción y se repite el ciclo. Las IRQ (Interrupt ReQuest) son líneas que llegan al controlador de interrupciones, un componente hardware dedicado a la gestión de las interrupciones, y que puede estar integrado en la CPU o ser un circuito separado conectado a la CPU.
- Un procesador principal (sin controlador de interrupciones integrado) suele tener una única línea de interrupción llamada habitualmente INT. Esta línea es activada por el controlador de interrupciones cuando tiene una interrupción que servir. Al activarse esta línea, el procesador completa la ejecución de la instrucción en curso y guarda el estado del programa en la pila.
- Después el procesador consulta los registros del controlador de interrupciones para averiguar qué IRQ es la que ha de atender. A partir del número de IRQ busca en el vector de interrupciones qué rutina debe llamar para atender la petición del dispositivo asociado a dicha IRQ.
- El vector de interrupciones es un vector que contiene el valor que apunta a la dirección en memoria de la rutina servidora de interrupción. En muchas arquitecturas de ordenadores los

vectores de interrupción se almacenan en una tabla en una zona de memoria, de modo que cuando se atiende una petición de interrupción de número n, el sistema transfiere el control a la dirección indicada por el elemento n-ésimo de dicha tabla.

- Otras maneras de ejecutar el gestor de la interrupción son las siguientes:
  - Cargar el contador de programa con un nuevo valor desde un registro específico o desde una posición de memoria.
  - Ejecutar la instrucción de llamada en una dirección proporcionada por un sistema externo.
  - Utilizar una señal de salida para reconocer la interrupción y tomar la instrucción de un dispositivo externo.
- Una vez finalizada la rutina servidora de interrupción, el procesador restaura el estado del programa interrumpido y vuelve al punto anterior a la interrupción.

4.3. Juego de instrucciones de la CPU.

4.3.1. Clasificación de los juegos de instrucciones.

- **CISC (Complex Instruction Set Computing)**. Se caracteriza por lo siguiente:
  - La CU es de lógica almacenada, y utiliza un número elevado de órdenes complejas que se dividen a su vez en otras más sencillas, de modo que una instrucción máquina se descompone en múltiples microinstrucciones. Sólo se puede ejecutar una instrucción cada vez y se necesitan varios ciclos de reloj para ejecutar una instrucción máquina completa.
  - El tamaño de las instrucciones es variable hasta los 64 bits, por lo cual el procesador debe realizar constantes accesos a memoria. El número de registros internos es reducido.
- **RISC (Reduced Instructions Set Computing)**. Se caracteriza por lo siguiente:
  - La CU es de lógica cableada, y utiliza un número reducido de órdenes simples, de modo que se necesitan más instrucciones para ejecutar una tarea. Al tratarse de instrucciones elementales, cada una se ejecuta en un ciclo de reloj. Esto permite la segmentación o *pipeline*, por la cual las instrucciones se recuperan en grupos. La CU examina cada grupo para comprobar si contiene instrucciones que pueden ejecutarse a la vez.
  - Todas las instrucciones tienen la misma longitud, normalmente 32 bits, y el número de registros internos es elevado, como mínimo 32. Por estos motivos el número de accesos a memoria del procesador es más reducido. Los modos de direccionamiento son sencillos, aunque siempre está presente el direccionamiento inmediato y el relativo a registro.

4.3.2. Tipos de instrucciones.

4.3.2.1. Según su funcionalidad.

4.3.2.1.1. Instrucciones de transferencia de datos.

- **LD m**. Carga [m] en AC.
- **LD #d**. Carga d en AC.
- **ST m**. Almacena [AC] en [m].
- **MOV m #d**. Almacena d en [m].
- **MOV m n**. Almacena [n] en [m].
- **PUSH**. Almacena [AC] en [PP] e incrementa PP.
- **POP**. Decrementa PP y almacena [PP] en AC.

- **IN e**. Carga en el AC el contenido del puerto de entrada e.
- **OUT s**. Almacena el contenido del puerto de salida s en AC.

4.3.2.1.2. Instrucciones de transferencia de direcciones

- **LEA**. Carga una dirección efectiva.
- **LDS**. Carga una dirección en el registro DS.
- **LES**. Carga una dirección en el registro ES.

4.3.2.1.3. Instrucciones aritméticas.

- **ADD m**. Almacena [AC] + [m] en AC.
- **SUB m**. Almacena [AC] - [m] en AC.
- **MUL m**. Almacena [AC] * [m] en AC.
- **DIV m**. Almacena [AC] / [m] en AC.
- **INC m**. Incrementa [m] en 1.
- **CLR m**. Pone todos los bits de [m] a 0.
- **SET m**. Pone todos los bits de [m] a 1.

4.3.2.1.4. Instrucciones lógicas.

- **CMP m**. Si [AC] < [m] se activa N, y si [AC] = [m] se activa Z.
- **AND m**. Almacena [AC] AND [m] en AC.
- **OR m**. Almacena [AC] OR [m] en AC.
- **XOR m**. Almacena [AC] XOR [m] en AC.
- **NOT m**. Almacena NOT [AC] en AC.

4.3.2.1.5. Instrucciones de desplazamiento.

- **SAL**. Desplazamiento aritmético a la izquierda.
- **SAR**. Desplazamiento aritmético a la derecha.
- **SHL**. Desplazamiento lógico a la izquierda.
- **SHR**. Desplazamiento lógico a la derecha.
- **ROL**. Rotación a la izquierda sin considerar el bit de acarreo.
- **ROR**. Rotación a la derecha sin considerar el bit de acarreo.
- **RCL**. Rotación a la izquierda considerando el bit de acarreo.
- **RCR**. Rotación a la derecha considerando el bit de acarreo.

4.3.2.1.6. Instrucciones de llamada.

- **CALL m**. Salva [CP] en la pila (dirección de vuelta) y pone [m] en CP.
- **RET**. Repone la dirección de vuelta de la pila en CP.
- **INT**. Llamada a interrupción
- **IRET**. Retorno de interrupción.

4.3.2.1.7. Instrucciones de control de flujo.

- **BR m**. Pone m en [PC].
- **BN m**. Pone m en [PC] si está activo N en PE.
- **BZ m**. Pone m en [PC] si está activo Z en PE.
- **BNZ m**. Pone m en [PC] si está activo N o Z en PE.

- **JMP**. Salto incondicional.
- **JG**. Salto si mayor.
- **JE**. Salto si igual.
- **JL**. Salto si menor.
- **JZ**. Salto si cero.
- **JNZ**. Salto si distinto de cero.

4.3.2.1.8. Instrucciones iterativas.

- **LOOP**. Bucle hasta el fin de una condición.
- **LOOPE**. Bucle mientras igual.
- **LOOPNE**. Bucle mientras distinto.
- **LOOPZ**. Bucle mientras igual a cero.
- **LOOPNZ**. Bucle mientras distinto de cero.

4.3.2.1.9. Otras instrucciones.

- **HALT**. Detener funcionamiento de la UCP hasta recibir interrupción.
- **NOP**. No operación, seguir con la siguiente instrucción.

4.3.2.2. Según su modo de direccionamiento.

- **Inmediato**. El valor del operando se indica en la propia instrucción de manera explícita.
- **Directo**. Se incluye la dirección de memoria en la que se encuentra el valor del operando.
- **Indirecto**. La instrucción incluye la dirección de memoria que contiene la dirección en la que se encuentra el valor del operando.
- **Relativo**. La instrucción incluye una dirección de memoria a la cual debe sumarse un índice ubicado en un registro para obtener la dirección en la que se encuentra el valor del operando.

4.3.2.3. Según su número de operandos.

- **Formato de cuatro direcciones**. Aparece en las primeras CPU e incorporan con la instrucción las direcciones de los dos operandos, del resultado y de la siguiente instrucción.
- **Formato de tres direcciones**. La CPU incorpora un contador de programa que se incrementa de manera automática al cargar una instrucción, haciendo innecesaria la inclusión de la dirección de la siguiente instrucción a ejecutar en cada instrucción. Este tipo de instrucciones incorporan las direcciones de los dos operandos y la dirección del resultado.
- **Formato de dos direcciones**. Utiliza sólo las direcciones de dos operandos, de manera que uno de ellos guarda el resultado de la operación una vez efectuada.
- **Formato de una dirección y media**. También utiliza las direcciones de dos operandos, pero en este caso uno de los operandos es un registro y necesita menos bits para direccionarse.
- **Formato de una dirección**. Uno de los operandos está cargado previamente en un registro conocido de la CPU, de modo que sólo es necesario indicar la dirección del otro operando.
- **Formato sin direcciones**. Ambos operandos están almacenados en registros conocidos de la CPU, y por tanto no es necesario indicarlo explícitamente.