3.1. INTRODUCCIÓN. ESTRUCTURA BÁSICA DE UN PROCESADOR

Volvamos por un momento a la estructura de la máquina de von Neumann, reorganizando ahora las unidades del computador desde un punto de vista más moderno, como el presentado en la figura 3.1.

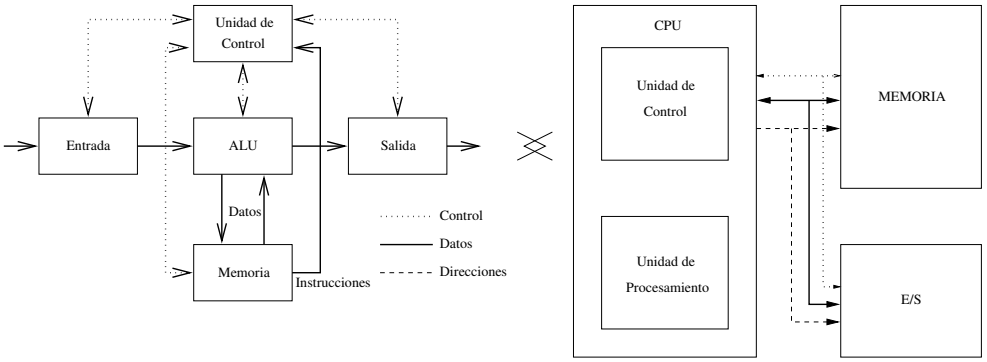

Figura 3.1: Unidades del computador

Como vemos, el computador se compone de los siguientes módulos:

1. **Unidad Central de Proceso (UCP o CPU)**. Compuesta por una unidad de control y otra de procesamiento (ALU más registros). Normalmente se construye sobre una pastilla de silicio que llamaremos procesador o microprocesador (μP).
2. **Memoria**. Subsistema en el que residen los programas y los datos. Este subsistema se organiza en varios niveles conformando lo que se conoce con el nombre de jerarquía de memoria: caché, memoria principal y memoria secundaria (disco duro).
3. **Unidad de Entrada/Salida**. Mediante la cual se controla la entrada y salida de información. Gestiona el manejo de periféricos en general: teclado, monitor, impresora, ratón, etc.

Estas tres unidades se encuentran interconectadas mediante buses de comunicación. Estos buses, no son más que líneas que transportan información binaria entre los diversos subsistemas. Los podemos clasificar en función del tipo de información que llevan:

Bus de direcciones. Mediante el cual la CPU selecciona la posición de memoria o la unidad de E/S de la cual va leer información o en la cual va a

escribir información. Evidentemente, si el bus tiene n líneas, tendremos acceso a 2^n posiciones distintas (a repartir entre RAM, ROM y E/S). Este bus es unidireccional.

Bus de datos. Mediante el cual se transfiere información de (hacia) la memoria o E/S hasta (desde) el procesador. Es un bus bidireccional y normalmente determina la palabra de trabajo del procesador (si el bus tiene n líneas decimos que el procesador es de n bits).

Bus de control. Mediante el cual se transfiere información de control entre el procesador, memoria y E/S. Es un bus bidireccional.

Podemos caracterizar un bus por el número de líneas de comunicación en paralelo (bits) que lo componen y por la frecuencia de transmisión de los datos en Hz. Ambos parámetros pueden en este caso fundirse en una sola magnitud, que actúa como métrica de su rendimiento, y que se conoce como *ancho de banda* (BW, del inglés "bandwidth"), indicando la velocidad de comunicación por el bus en bits por segundo (bps.). Por ejemplo, el bus local PCI del Pentium tiene una anchura de 64 líneas y una frecuencia de 66 MHz., lo que proporciona un ancho de banda de 528 Mbytes/s.

En este tema nos centraremos en el procesador central de forma general. Es el componente más complejo del sistema y su misión es sincronizar y coordinar el funcionamiento de los demás módulos del sistema, leyendo, decodificando y ejecutando instrucciones.

3.1.1. Características principales del procesador

1. **Longitud de palabra.**

   Definida por el n.º de bits del bus de datos, bus de direcciones, tamaño de los registros o por el n.º de bits con que la ALU puede operar en paralelo. En los procesadores modernos la longitud de palabra impone un limite máximo al rango de direccionamiento de memoria.

   Normalmente los registros de la máquina tienen la misma anchura, pero no tiene porqué ser así. Por ejemplo, el 68000 de Motorola tiene registros de 32 bits, pero tiene un bus de datos de 16 bits. El Intel 8088 tiene registros de 16 bits y un bus de 8 bits.

   De cualquier modo, el que la longitud de palabra sea 32 bits (por ejemplo), no quiere decir que toda la información en este μP particular ocupe 32 bits. Es decir, podemos codificar caracteres con 8 bits, códigos de operación con 16 bits y números en punto flotante de doble precisión con 64 bits.

   En general nos referiremos con *longitud de palabra del procesador* al

número de bits de los registros accesibles al programador (anchura de la ALU).

2. **Velocidad de proceso.**

   Estará en función de ciertos parámetros de diseño:

   - Frecuencia de reloj (33MHz, 100MHz, 200MHz, etc)
   - Tiempo de ejecución de cada instrucción: CPI (ciclos de reloj por instrucción)
   - Tiempo de acceso a memoria y a E/S. Normalmente estas dos unidades ralentizan en gran medida al sistema completo.

   El tiempo que un procesador invierte en la ejecución de un programa puede obtenerse como el producto de tres factores:

   a) NI: Número de instrucciones máquina en que se transforma el programa.
   b) CPI: Número medio de ciclos de reloj que se necesitan para ejecutar cada instrucción máquina.
   c) T: Tiempo de ciclo de reloj (o su frecuencia f como magnitud inversa).

   Es decir, $T_{CPU} = NI \times CPI \times T = \frac{NI \times CPI}{f}$, donde T_{CPU} resultará en la misma magnitud en que expresamos el período T, por ejemplo, en microsegundos si la frecuencia f viniera expresada en MHz.

   El primero de esos factores depende principalmente del compilador. Si éste está optimizado para un procesador, será capaz de traducir un programa de alto nivel (escrito en C, por ejemplo) en un programa objeto o ejecutable que contenga el menor número de instrucciones máquina posibles.

   El segundo factor, CPI, está más relacionado con el conjunto de instrucciones del procesador. Por ejemplo, un procesador RISC, (cuyo conjunto de instrucciones se compone de pocas instrucciones muy sencillas) puede reducir el CPI incluso a la unidad, pero a costa de incrementar NI. Para ello, una de las técnicas que utiliza es la de *segmentar* ("pipeline") la ejecución de instrucciones dentro del procesador.

   El tercer factor es la frecuencia de reloj del procesador, que está más relacionada con su tecnología hardware y con las técnicas de integración de circuitos. La positiva evolución de éstas ha permitido duplicar la frecuencia del procesador cada 2 ó 3 años en las dos últimas décadas.

3. **Capacidad de proceso.**

   En relación con:

   a) Número de instrucciones del μP, distinguiendo entre los del tipo

CISC (Complex Instruction Set Computer) y RISC (Reduced Instruction Set Computer). Con relación a este parámetro se utiliza la métrica MIPS (millones de instrucciones por segundo) que viene definida como:

$$MIPS = \frac{NI}{T_{CPU}} \cdot 10^{-6}$$

La principal ventaja de los MIPS es que son fáciles de comprender. Sin embargo, son dependientes del repertorio de instrucciones, pues a mayor complejidad de las instrucciones máquina de un procesador, menor número de MIPS producirá un programa concreto, sin que eso indique que el programa se está ejecutando más lentamente.

b) Operaciones que puede realizar la ALU (aritmética entera, BCD, formato en punto flotante de simple o doble precisión, división y multiplicación por HW o por SW, etc...). Los MFLOPS o millones de operaciones en punto flotante por segundo (del inglés *Million of FLoating-point Operations Per Second*), caracteriza la velocidad de la unidad en punto flotante y por otro lado es una métrica que está basada en operaciones en lugar de instrucciones como los MIPS.

4. **Gestión de memoria.**

Se refiere a las características que posee el μP en relación con la gestión de memoria. Por ejemplo, si incluye HW para el manejo de caché, si lleva un nivel de caché interno, si implementa una unidad de manejo de memoria (MMU) para gestionar la memoria virtual, etc.

Los dos parámetros principales que influyen en las prestaciones de una memoria son su *tamaño* en número de palabras y su *latencia* o tiempo de respuesta expresado normalmente en nanosegundos (ns.). El tamaño aumenta la funcionalidad del sistema al permitir ejecutar aplicaciones más complejas, mientras que la latencia está más ligada a la rapidez de ejecución de los programas. El número de bytes que la memoria sea capaz de proporcionar por unidad de tiempo, es decir, su *ancho de banda* vendrá dado como:

$$BW = \frac{longitud\_de\_palabra\_de\_memoria}{latencia}$$

Capacidad y latencia son parámetros inversamente ligados con respecto al precio, es decir, a mayor tamaño, menor rapidez por el mismo precio. Por ejemplo, las memorias caché actuales tienen un tamaño de

256-512 Kbytes y una latencia de unos pocos ns., la memoria principal o RAM presenta un tamaño de entre 16 y 128 Mbytes y una latencia de decenas de ns., y la memoria secundaria o disco duro llega a varios Gbytes, pero sube la latencia a los milisegundos.

5. **Manejo de interrupciones y capacidad de interfaz.**
   Se refiere a las capacidades del μP para relacionarse con otros dispositivos externos. Por ejemplo, como maneja las interrupciones, si soporta dispositivos de acceso directo a memoria (DMA) para acelerar la transferencia de datos entre periféricos y memoria, etc.

3.2. SUBSISTEMAS DE DATOS Y DE CONTROL

Como hemos repetido en otras ocasiones, el procesador engloba las unidades de control, de datos y los registros. En este apartado daremos una primera introducción a estos tres bloques:

3.2.1. Registros

Los registros son dispositivos digitales que nos permiten almacenar información y acceder a ella en tiempos bastante menores que el que necesitaríamos para acceder a memoria. Podemos clasificar los registros que podemos encontrar en un μP en función de la información que almacenan:

1. **Registros de propósito general (RPG):** En estos registros almacenamos la información que utilizamos más frecuentemente (ya que el acceso a memoria es más lento). Podemos distinguir dos subtipos de RPG's:
   - Registros de datos: Almacenan datos y tienen normalmente un número de bits igual al de la palabra del μP.
   - Registros de direcciones: Normalmente almacenan direcciones, por lo que deben tener un tamaño igual al del bus de direcciones.
2. **Contador de Programa (PC o IP):** Contiene la dirección de memoria de la cual se leerá la siguiente instrucción a ejecutar. Normalmente el contenido del PC se incrementa al ejecutar cada instrucción para que "apunte" a la siguiente instrucción a ejecutar. Si modificamos el contenido del PC cargándolo con la dirección x, estamos realizando un "salto" a la instrucción almacenada en la posición x de memoria.
3. **Registro de Instrucciones (RI):** Almacena el código de operación de la instrucción que estamos ejecutando en un momento dado. Es un

registro "transparente al usuario", es decir, el usuario o el programador no pueden acceder a ese registro y modificar su valor, sino que es un registro que actualiza automáticamente la sección de control del μP.

4. **Acumuladores:** En muchos μP hay uno (o varios) registros acumulador (AC) en el que implícitamente hay un operando y donde se "acumula" el resultado. En caso de no tener registro AC, podemos utilizar registros de propósito general.
5. **Registro índice y registro base:** Pueden ser registros de direcciones que se utilizan para el direccionamiento relativo a base (el registro base contiene la dirección base a cierta posición de memoria) o para el direccionamiento indexado (el registro índice va apuntando a las distintas posiciones de una estructura de datos con almacenamiento consecutivo: arrays, cadenas alfanuméricas, etc).
6. **Puntero de Pila (Stack Pointer -SP-):** Puede ser un registro de dirección que contiene el puntero a la posición de la pila escrita más recientemente (la cima de la pila).
7. **Registros temporales:** Normalmente incluidos dentro del μP para guardar resultados intermedios de algunas operaciones (por ejemplo, la instrucción `XCHG`, que intercambia el contenido de dos registros necesita de un registro temporal para realizar el intercambio). Estos registros no suelen ser accesibles por el usuario (son transparentes al usuario).
8. **Registro de estado o de banderas (Flags Reg. o SR):** Es un registro en el que cada bit o campo de bits tienen información independiente, normalmente relacionada con el resultado de las operaciones realizadas en la ALU. A cada uno de esos bits independientes se le llama bandera (flag) y se activan o desactivan en función de la ejecución de ciertas instrucciones. Estos flags son testeados o chequeados por otras instrucciones para realizar saltos condicionales.

   Los flags más comunes son los siguientes:

   a) Cero (Z): Se pone a 1 si el resultado de una operación es 0.
   b) Carry (C): Se pone a 1 si el resultado de una operación provoca un acarreo de salida.
   c) Signo (S): Se pone a 1 si el resultado de una operación es negativo.
   d) Overflow (O): Se pone a 1 si el resultado de una operación se ha salido del rango de valores representables.
   e) Paridad (P): Se pone a 1 si el resultado de una operación da lugar a una palabra con un número par de 1's (o un número impar de 1's, dependiendo del convenio).

Otros flags no son el resultado de realizar operaciones con datos, sino que son activados o desactivados por el programador para obligar al μP a trabajar en uno de varios modos posibles:

a) **Habilitación/Deshabilitación de interrupciones:** El programador pone a 1 este flag si quiere permitir las interrupciones de otros dispositivos. En caso de no aceptar interrupciones, éstas pueden ser enmascaradas poniendo a 0 este flag.
b) **Traza:** A 1 habilita la ejecución de instrucciones paso a paso (modo de depuración de programas).
c) **Supervisor:** A 1 el procesador trabaja en modo supervisor (modo de alta prioridad en el que se pueden ejecutar instrucciones privilegiadas y no hay limitaciones en el acceso a datos -es el modo en que trabaja el Sistema Operativo-). A 0 el procesador trabaja en el modo normal o modo de usuario.

3.2.2. Unidad de Datos

También llamada Sección de Procesamiento, donde el módulo principal es la Unidad Aritmético-Lógica. Este módulo (ALU) contiene la circuitería necesaria para realizar las operaciones aritméticas y lógicas. Normalmente la ALU sólo realiza operaciones enteras, por lo que también se la conoce con el nombre de "unidad entera" para diferenciarla de la "unidad en punto flotante" o FPU[4].

La unidad entera, también es la encargada de calcular la dirección efectiva del operando (ya que en los direccionamientos relativos es necesario realizar sumas o restas para determinar la dirección del operando).

El número de operaciones que puede realizar la ALU y la velocidad de procesamiento dependen en gran medida de la cantidad de HW que utilicemos para implementar las distintas funciones. Podemos abaratar costes en el desarrollo de la ALU emulando ciertas operaciones mediante SW, de forma que no sea necesario incluir un HW específico para esas operaciones. Sin embargo, el tiempo de ejecución de estas instrucciones emuladas mediante SW es mayor.

Como vemos en la figura 3.2 la ALU se alimenta de los registros o de memoria (a través del bus de datos). La salida de la ALU también puede almacenarse en registros o en memoria. También debe existir una entrada de control que indique a la ALU que operación debe realizar (en el caso de la

[4]La FPU también puede estar incluida dentro del μP o colocarse externamente como coprocesador matemático.

figura, las señales c6, c7 y c8 codifican la operación que debe realizar la ALU). Además, la ALU debe actualizar el registro de Estados (SR) después de cada operación.

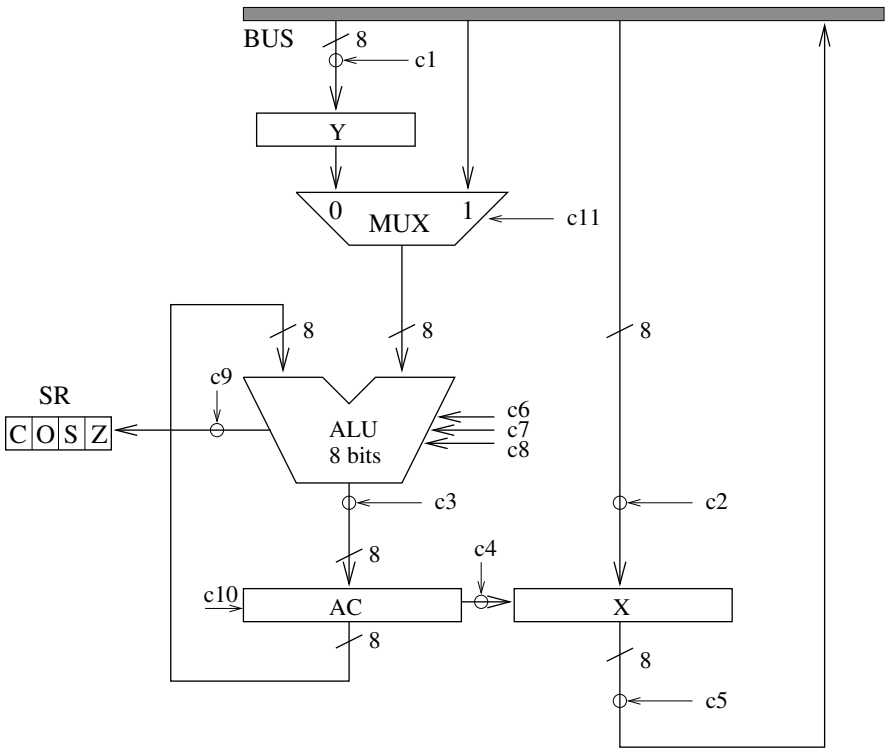

Figura 3.2: Una posible Sección de Procesamiento

Como vemos en la figura 3.2, esta sección de procesamiento tiene varios **"puntos de control"** (de c1 a c11) que deben ser activados adecuadamente para que el secuenciamiento de las operaciones dé lugar a una ejecución correcta de las instrucciones.

El modo particular en el que interconectamos los registros, la ALU, el SR, etc, determina lo que llamaremos el **"flujo de datos"** *(Data Path)*. Este flujo de datos determina cuáles son los posibles caminos que puede tomar la información durante su procesado.

3.2.3. Unidad de Control

También llamada Sección de Control. Es la unidad que genera las señales de control que "atacan" los puntos de control de la sección de procesamiento. Por tanto, esta unidad debe tener la circuitería de control, temporización, y decodificación de instrucciones.

En la figura 3.3 podemos ver un esquema a alto nivel de una sección de control. A partir del contenido del registro de instrucciones, la sección de control debe generar las señales de control para que se ejecute dicha instrucción adecuadamente.

Después del ciclo de búsqueda (fetch), en el registro de instrucción encontraremos la instrucción que vamos a ejecutar. El código de operación de esta instrucción debe ser decodificado durante la fase de decodificación, de forma que la sección de control interprete el objetivo de la instrucción en curso. Una vez decodificada la instrucción, debemos ejecutarla generando de forma secuencial las señales de control que activan las distintas funciones de la sección de procesamiento.

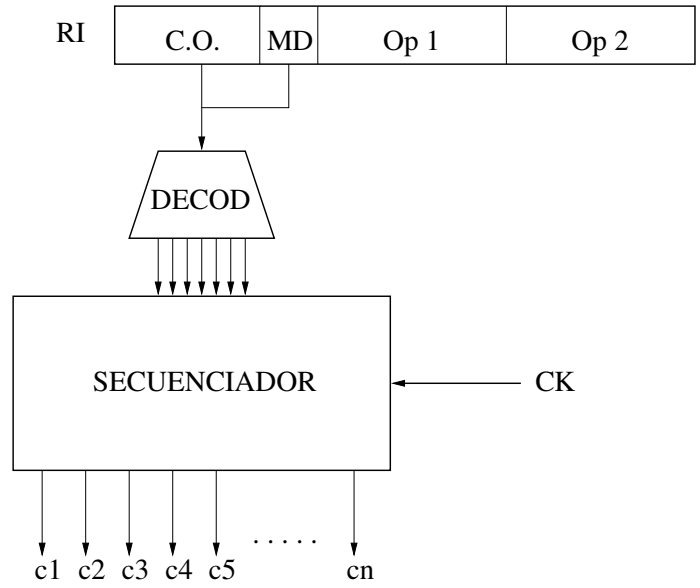

Figura 3.3: Esquema a alto nivel de una sección de control

Para conseguir este objetivo, la sección de control contiene un decodificador y un secuenciador (una máquina de estados secuencial alimentada por una

señal de reloj). El decodificador decodifica el código de operación en curso y genera un conjunto de señales que inicializan el secuenciador colocándolo en un estado inicial. Este estado inicial será distinto para las distintas instrucciones a ejecutar. A partir de este estado inicial el secuenciador evolucionará a través de ciertos estados, cada uno de los cuales generará ciertas señales de salida (señales de control) que activarán los distintos puntos de control de la sección de procesamiento.

A cada conjunto de señales de control para un estado dado se le llama **microinstrucción**, y por tanto la ejecución de cada instrucción máquina (**macroinstrucción**) se traduce en un conjunto de micro-instrucciones. El conjunto de microinstrucciones que ejecutan una determinada macroinstrucción se conoce con el nombre de **microsubrutina**.

Existen básicamente dos técnicas para construir el secuenciador:

1. **Técnica cableada:** El secuenciador se construye mediante una máquina de estados compuesta de puertas, contadores, registros, decodificadores, etc. La interconexión de estos elementos es un poco heurística, poco sistemática y da lugar a secciones de control complicadas, pero bastante rápidas. Es la técnica de control utilizada en los procesadores RISC.
2. **Técnica microprogramada:** El secuenciador se construye mediante un contador de microprograma que apunta a las distintas microinstrucciones almacenadas en una memoria (llamada micro-memoria). Esta técnica es más sistemática y pedagógica, pero da lugar a secciones de control más lentas. Es la técnica utilizada en los procesadores CISC.

3.3. CICLO MÁQUINA Y ESTADOS DE UN PROCESADOR

Como hemos comentado en alguna otra ocasión, la ejecución automática de un programa por un computador se lleva a cabo mediante el procesado de las instrucciones contenidas en el programa. A su vez, la ejecución de estas instrucciones se efectúa por el uso repetitivo de ciertas operaciones del sistema, a saber:

Búsqueda de instrucción. El procesador "busca" en la posición de memoria apuntada por el registro PC una instrucción que carga en el registro RI.

Decodificación. El C.O. de la instrucción se analiza para determinar que función debe realizar la instrucción. En términos más realistas, podemos decir que el C.O. decodificado proporciona un estado inicial a la unidad

de control, de forma que ésta genere la información de control necesaria para ejecutar esta instrucción en particular. También en esta fase se incrementa el PC para que apunte a la siguiente instrucción a ejecutar.

Cálculo de la dirección efectiva. En caso de que la instrucción necesite operandos almacenados en memoria, se determina la dirección en la que se encuentran dichos operandos: la dirección efectiva o EA (*efective address*).

Búsqueda de operandos. Acceso a memoria para leer los operandos de la instrucción.

Ejecución. Realización de la función especificada por la instrucción. Una vez concluida esta fase volvemos a la fase de búsqueda de la siguiente instrucción.

Antes de entrar en este bucle de ejecución de instrucciones hay que inicializar el computador para ejecutar un determinado programa. Para ello el sistema operativo debe:

1. Cargar el programa y los datos en memoria
2. Inicializar el PC para que apunte a la primera instrucción del programa a ejecutar.

Llegados a este punto podemos formular las siguientes definiciones:

Ciclo de instrucción. Tiempo necesario para leer y ejecutar una instrucción. El ciclo de instrucción es mayor para instrucciones más lentas y menor para instrucciones más rápidas. Normalmente el ciclo de instrucción se mide en ciclos de reloj. Por ejemplo, en el 8086 de Intel la instrucción ADD necesita 3 ciclos de reloj para completarse; la instrucción MUL necesita de 70 a 77 ciclos de reloj.

Ciclo máquina. Subperiodos en que se divide un ciclo de instrucción. Estos subperiodos pueden necesitar también uno o varios ciclos de reloj. En el tema siguiente veremos en detalle en qué consiste el ciclo máquina.

3.3.1. Modos de secuenciamiento

Nos referiremos aquí a la secuencia en que se pueden ejecutar las instrucciones dentro de un programa. Básicamente podemos distinguir cuatro modos:

1. **Ejecución secuencial:** Las instrucciones se ejecutan de forma secuencial incrementándose el PC sin discontinuidades. Es decir, se ejecutan en orden las instrucciones almacenadas consecutivamente en memoria.
2. **Salto** (transferencia de control): Mediante las instrucciones de salto el PC se puede cargar con una dirección distinta a la siguiente. De esta

forma se transfiere el control a otro punto del programa. El valor previo del PC (antes del salto) se pierde al modificar el contenido de éste.

3. **Subrutina** (transferencia de control): El valor del PC se modifica de forma que se provoca una transferencia de control a otro punto del programa (donde comienza una subrutina). La diferencia con el **salto** estriba en que el valor del PC (antes de su modificación) se "salva" en la pila. Para llamar a la subrutina se utiliza la instrucción **call**. La subrutina termina con una instrucción **ret** (de *return*) de forma que se recupera el valor del PC almacenado en la pila y la ejecución sigue por la instrucción siguiente al **call**.
4. **Interrupción** (transferencia de control): Mediante este modo también se produce una transferencia de control a una subrutina (llamada ahora subrutina de tratamiento de interrupción), pero, en general, dicha transferencia no la provoca el programador, sino que puede ocurrir en cualquier momento de la ejecución del programa. Por tanto, en la pila no debemos salvar únicamente el PC, sino también el resto del "estado de la máquina" (registro de estados, registros de propósito general, etc.). Al terminar la rutina de tratamiento de interrupción recuperamos el PC y el estado de la máquina y continúa la ejecución del programa.

3.3.2. Técnica pipeline de instrucciones

La técnica de segmentación o pipeline consiste en dividir una tarea en subtareas y ejecutar cada una de estas subtareas mediante una unidad funcional independiente. Por ejemplo, la tarea de fabricación de un coche se puede dividir en varias subtareas y ejecutar cada una de ellas por un operario dentro de una cadena de montaje. Esta estrategia se puede aplicar dentro de un procesador para acelerar la ejecución de instrucciones.

Por ejemplo, el 8086 de Intel, divide la tarea de ejecutar una instrucción en dos subtareas:

1. Búsqueda de instrucción
2. Decodificación, cálculo de EA, búsqueda de op. y ejecución

El 8086 dedica una unidad llamada BIU (unidad de interfaz del bus) a la búsqueda de instrucciones y otra unidad independiente, EU (unidad de ejecución), a la ejecución de instrucciones. Las dos unidades operan en paralelo, de forma que mientras se ejecuta la instrucción actual, ya se está buscando la instrucción siguiente en paralelo. De esta forma conseguimos reducir el tiempo de ejecución. Este caso particular de pipeline de dos etapas se conoce con el nombre de *prefetch* (prebúsqueda).

En general, podemos utilizar tantas unidades funcionales como subtareas queramos ejecutar. Por ejemplo, podríamos haber dedicado una unidad funcional para cada una de las 5 fases: búsqueda de instrucción, decodificación, cálculo de la EA, búsqueda de op. y ejecución; de forma que, en el caso ideal, el tiempo de ejecución de un programa se divide por 5.

3.4. EJEMPLO: MICROPROCESADOR 8086

3.4.1. Características generales

El 8086 y 8088 son microprocesadores de Intel de propósito general. Ambos son prácticamente idénticos excepto por el tamaño de su bus de datos externo. En el caso del primero, el bus de datos es de 16 bits, mientras que en el segundo, es de 8 bits. La razón para crear el 8088 con este bus de datos reducido fue prever la continuidad entre el 8086 y los antiguos procesadores de Intel (8085). En este ejemplo nos centraremos en el 8086.

Resumen de las características:

- Microprocesador de 16 bits (ancho del bus de datos)
- Bus de direcciones de 20 bits (máxima memoria direccionable: 1 MegaByte).
- E/S no mapeada en memoria. El espacio de direcciones de E/S es independiente del de memoria. Tamaño del espacio de E/S es de 64Kbytes
- Encapsulado de 40 pines. Algunas señales multiplexadas.
- Segmentación. Dos etapas: B.I.U (unidad de interfaz con el bus), que realiza la búsqueda *(prefetch)*, y E.U (unidad de ejecución), que recoge las instrucciones de la cola de la B.I.U. y las ejecuta.

3.4.2. Registros

El 8086 presenta un conjunto de 14 registros de 16 bits que se pueden agrupar de la siguiente forma:

Registros generales : Su función es el almacenamiento temporal de datos.

AX (acumulador). Es utilizado en las instrucciones aritméticas.
BX (base). Se usa generalmente para indicar un desplazamiento.
CX (contador). Se utiliza en bucles.
DX (datos). Se utiliza también en operaciones aritméticas.

Estos registros ofrecen la posibilidad de usarse además como registros de 8 bits, refiriéndose al byte más significativo con AH, BH, CH y DH o al menos significativo con AL, BL, CL y DL respectivamente.