INTRODUCCION A LAS COMPUTADORAS

TEMA 1. INTRODUCCION

1.1 LOGICA PROGRAMADA FRENTE A LOGICA CABLEADA

Ventajas de la lógica programada

Bajos costes, más flexibles, mayor facilidad de diseño y capacidad de realizar tareas muy complejas.

Desventaja: Es mucho más lenta frente a la lógica cableada.

1.2 DEFINICION DE COMPUTADOR

Máquina de propósito general que procesa datos de acuerdo con el conjunto de instrucciones almacenadas internamente temporal ó permanentemente.

Proceso de datos: Capacidad de calcula, comparar, copiar y transferir.

1.3 EVOLUCION HISTORICA

1ªGeneración

El primer computador fue el UNIVAC I que almacenaba 1000 palabras de 12 bits, funcionaban con válvulas de vacío. Tenían un gran consumo y eran de gran tamaño.

2ª Generación

Hacia finales de los 50. Estas computadoras funcionaban con transistores. Hay un aumento de la fiabilidad. La memoria estaba compuesta por núcleos de ferrita. Un ejemplo es el IBM 1401 y el Honeywell 800.

3ª Generación

Hacia mediados de los 60. Estos utilizan ya circuitos integrados. Aparecen los CI's: SSI y MSI (Small / Medium Scale Integration), los primeros sistemas operativos y los gestores de bases de datos. Comienzan a implantarse en grandes empresas. Un ejemplo es el IBM 360.

4ª Generación

Hacia finales de los 70. Aparecen los CI's: LSI y VLSI (Large / Very Large Scale Integration). Aparece el microprocesador y el ordenador se introduce en la pequeña empresa y a nivel doméstico.

5ª Generación

Hacia principios de los 90. Aparecen los ordenadores multimedia y los sistemas operativos amigables con entornos gráficos para facilitar la utilización a usuarios no especialistas. Se reducen costes y tamaño y aumenta la capacidad de proceso, la memoria y la velocidad.

1.4 LA ARQUITECTURA VON−NEUMANN

Es la arquitectura en la que se basan los primeros ordenadores y es en la que se basa el 90% de los ordenadores actuales. John Von−Neumann era un matemático húngaro (1903−1957). En 1945 se le ocurre el diseño de una máquina compuesta por la CPU, una memoria principal y un interface de E/S, todos ellos interconexionados con un bus de interconexión. Sólo tiene un único flujo de datos e instrucciones. El introduce la idea de programa almacenado y la idea de ruptura secuencial. Hasta entonces las instrucciones de introducían una a una y se ejecutaban secuencialmente según entraban. Cuándo se introducía un programa a ejecutar había que introducirlo entero y junto con él todos los datos y era imposible alterar la secuencia de órdenes. Hasta aquí no se almacenaba nada en memoria. Con Von−Neumann se guarda el programa en memoria y se introducen los datos cuándo se ejecuta el programa. Dado que todas las instrucciones están disponibles, la siguiente instrucción a ejecutar puede ser cualquiera.

instrucciones

UNIDAD DE MEMORIA PRINCIPAL

CONTROL

Datos

Datos

ALU Datos

UNIDAD DE ENTRADA Datos

Datos UNIDAD DE SALIDA Datos

CPU

1.4.1 MEMORIA PRINCIPAL

Su destino es contener el programa en ejecución y contener parte de los datos del programa.

1.4.2 UNIDAD ARITMETICO−LOGICA

Es una de las partes de la CPU. Realiza operaciones aritméticas y lógicas y lo que entrega es el resultado de las operaciones y además algunas particularidades del resultado (desbordamiento, signo, valor=0, ...)

1.4.3 UNIDAD DE CONTROL

Es la otra parte de la CPU. Se encarga de coordinar que todos los procesos se realicen de manera ordenada y secuencial. Su modo de proceder contiene un puntero que indica la siguiente instrucción a ejecutar, obtiene la instrucción de memoria, la descodifica (serie de microórdenes internas) y repite el proceso.

1.4.4 UNIDADES DE ENTRADA / SALIDA

Están interconectadas con la CPU y con el exterior mediante líneas dedicadas ó mediante líneas de propósito general.

1.4.5 BUSES y PUERTAS TRIESTADO

Los buses son los interconectores que interconectan todas las unidades anteriores.

Las puertas triestado tienen un nivel alto, un nivel bajo y un nivel de alta impedancia.

El bus simplifica mucho el diseño interconexión. El tamaño del bus es un identificador de la capacidad de proceso del ordenador.

TEMA 2. MEMORIAS

2.1 DEFINICION DE MEMORIA

Dispositivo ó conjunto de dispositivos capaces de almacenar información. Debe disponer de un sistema de recuperación de información y de un sistema de direccionamiento de la misma.

Punto de memoria: Lugar físico dónde se almacena un bit.

2.2 CARACTERISTICAS DE LA MEMORIA

2.2.1 CAPACIDAD

Cantidad de información que puede almacenar en una memoria en concreto. Hay dos tipos: *Util: LA que puede almacenar el usuario.

*Bruta: Número total de unidades de información (bits ó bytes) que pueden ser

almacenadas en un dispositivo.

2.2.2 VELOCIDAD

Indica el tiempo desde que se solicita un dato hasta que se recibe. Dos grupos:

* De acceso aleatorio: Si se tarde lo mismo independientemente de la situación del dato.

- De acceso no aleatorio: Influye el tiempo de búsqueda y el de latencia.

2.2.3 DURACION DE LA INFORMACION

Capacidad de retener la información a lo largo del tiempo. Tiene que ver con la permanencia ó no del suministro de energía eléctrica.

*Volátiles: Pierden su contenido sin electricidad. Están basadas en semiconductores.

*No volátiles: Permanentes.

2.2.4 COSTES

Se divide en precio/unidad de información y depende fuertemente de forma antagónica con la velocidad de acceso.

Precio T.acceso Coste

Registros 20000 pts/MB 10 ns

RAM 5000 pts/MB 70 ns

Disco duro 100 pts/MB 10 ms

Disquetes 50 pts/MB 2 s

Cintas Streamer 10 pts/MB 20 s

Papel digital 0,01 pts/MB 2 min. Tiempo

2.2.5 MODO DE ACCESO

Se refiere a como se indica la dirección de la posición de memoria a la que se quiere acceder y como se organiza la información en el interior de la memoria.

2.2.6 JERARQUIAS DE MEMORIA

Lo que interesa es utilizar memorias rápidas, pero son muy costosas, por lo que se utilizan dispositivos caros(rápidos) para los que operan en la CPU y dispositivos baratos(lentos) pero con mayor capacidad de almacenar información que no se procesa en ese momento determinado.

ns REGISTRO En la CPU

decenas de ns MEMORIA CACHE

MEMORIA PRINCIPAL Placa base

de ms a seg. MEMORIA SECUNDARIA Fuera del ordenador

2.3 CLASIFICACION DE MEMORIAS

Acceso directo:

Sólo lectura(ROM): ROM, PROM, EROM, EEROM, EAROM

Lectura/escritura(RAM): Estáticas y dinámicas.

Acceso asociativo(CAM)

Acceso BORAM: CCD, Burbujas magnéticas.

Acceso serie: Pilas FIFO, Pilas LIFO

Acceso directo ó cíclico: Unidades de disco, unidades de tambor

Acceso secuencial: Unidades de cinta

2.3.1 MEMORIAS DE ACCESO DIRECTO

Construidas a base de semiconductores. Dos grupos:

Lectura:

ROM: en ellas los datos se almacenan en el momento de fabricación del CI.

PROM: Se llaman programables ROM, que son CI en los que el punto de memoria reside en un fusible. Para grabar los datos se funden los fusibles necesarios. Sólo admiten una grabación.

EPROM: Erasable, programable ROM. Son memorias en las que se pueden realizar varios ciclos de borrado y grabación. Para borrar se ilumina con luz ultravioleta el interior del CI por una ventana que tiene. Es necesario borrarla para introducir datos.

EEROM: Electrical, erasable ROM. El borrado se realiza eléctricamente.

EAROM: Electrical alterable ROM. Se pueden alterar los datos sin borrado previo.

Lectura/Escritura:

Son las memorias RAM (Random Access MEmory).Dos tipos:

Estáticas: Utilizan biestables. LA información permanece con la alimentación. Tienen grandes velocidades de acceso. El sistema de control y de conexión a buses es muy sencillo. La capacidad de integración no es muy alta.

Una variante de estas memorias es la **VRAM** que es la RAM de vídeo, la cual permite el acceso a los datos por dos caminos diferentes y simultáneamente, se llama también de doble puerto. La información que tiene es la de presentación en pantalla. Los accesos que tiene son de tipo secuencial y continuo por el controlador gráfico y cada vez que se modifica la imagen, la CPU accede a cambiar datos. La VRAM elimina el doble acceso pero es bastante más cara.

Dinámicas: Utilizan condensadores que es una capacidad parásita que tienen los transistores MOS. La escala de integración es muy alta. La velocidad de acceso es relativamente baja; 60 ns. EL control de los CI's es complejo y se complica por que hay que renovar la información, es decir, hacer un refresco de memoria cada poco tiempo (entre 2 y 4 msg.)

−.En función de como se hace dicho refresco se dividen en:

DRAM: RAM Dinámica estándar. La circuitería que realiza el refresco es externa al CI.

PSRAM: Pseudo Static RAM. En el CI está gran parte de la circuitería de refresco. Lo único que queda fuera es una línea externa denominada CK, que marca el momento de iniciar el refresco.

VSRAM: Virtually Static RAM. Internamente gestiona todas las tareas de refresco, por tanto, a efectos prácticos es como si se tratara de una RAM estática.

2.3.2 MEMORIAS DE ACCESO ASOCIATIVO

CAM: Contents Access Memory. Memoria de acceso por contenidos. Tiene dos modos de acceso:

*Aleatorio: Que es el normal y que se utiliza generalmente para las operaciones de escritura. Con este tipo de acceso se da una dirección y se devuelve el dato de esa dirección.

*Asociativo: Se le pregunta a la memoria si el dato solicitado está en alguna posición de memoria. Si el dato está escrito en alguna dirección, devuelve la dirección y un indicador de encontrado ó no. Para hacer esto hay que tener un comparador en memoria para cada posición con lo que se hace muy costosa y voluminosa.

2.3.3 MEMORIAS DE ACCESO BORAM

BORAM: Block Oriented RAM (Memoria de Acceso aleatorio y orientado a bloques)

Se busca el bloque en el que está el dato, se posiciona en dicho bloque y transfiere el dato secuencialmente; con lo que se parece a los discos. Se diferencia de éstos en que el acceso al bloque es aleatorio.

−*CCD:* Memorias de acoplamiento de carga. (Charge Coupled Device).

−Memorias de burbujas magnéticas.

2.3.4 MEMORIAS DE ACCESO SERIE

Se realizan mediante biestables de semiconductor. Son grandes registros de desplazamiento. Dependiendo de como se recuperen los datos, existen dos tipos:

FIFO: (First In, First Out), que son las colas.

LIFO: (Last In, First Out), que son las pilas.

Una aplicación típica de las pilas es almacenar las direcciones de retorno de subrutinas.

2.3.5 MEMORIAS DE ACCESO DIRECTO ó CICLICO

Permiten acceder directamente a un bloque de datos y dentro de ese bloque se accede al dato de forma secuencial. Ejemplo: los discos.

2.3.6 MEMORIAS DE ACCESO SECUENCIAL

Para acceder a un dato hay que pasar por todas las posiciones físicas anteriores. Ejemplo: las cintas.

2.4 CHIPS DE MEMORIA DE ACCESO ALEATORIO

2.4.1 ESTRUCTURA / ORGANIZACION DE LAS RAM

Bloques en los que se divide una RAM:

−.− El registro de dirección S más el decodificador asociado.

−.− Los circuitos de control que coordinan las acciones de lectura/escritura.

−.− La matriz de memoria.

−.− El registro de palabra M.

2.4.2 COMO SE ACCEDE A MEMORIA

6

Bus interno

1 2 3 4

CS **5**

WE

CS

WE **5 6** Bus de datos

1. Bus de direcciones.

2. Registro de direcciones S.

3. Decodificador. 7

4. Matriz.

5. Circuitos de control.

6. Registro de palabra M.

7. Bus de datos.

Un 0 en CS inhabilita el acceso, ya sea para lectura como para escritura.

Si CS=1 se habilita el acceso. WE indica si es lectura ó escritura.

Un circuito RAM tiene el registro de palabra bidireccional, si es lectura irá de la matriz al registro, y si es escritura, al revés.

2.4.3 TIPOS DE ORGANIZACION DE LA MATRIZ DE MEMORIA

2D

Cada punto de memoria se selecciona con una línea. El decodificador de memoria es muy grande y el punto de memoria muy sencillo. Es el dibujo de la página anterior.

Ejemplo: Con direcciones de 16 bits, se necesitan 216=65536 conductores en el decodificador. **1**

3D

2

3

SF 4

7

5 1.Palabra: el mismo bit de cada capa

2.Registro de palabra M.

SC 3.Bus de datos.

4.Decodificador de filas.

5.Decodificador de columnas.

6.Bus de direcciones.

6 7.Circuitos de control.

Cada punto de memoria se selecciona mediante dos líneas. El registro S se divide en dos partes, cada una con un decodificador independiente. Se reduce notablemente el número de líneas de los decodificadores. Existen tantas capas de memoria como bits tenga la palabra. Con el mismo ejemplo que en 2D, una dirección de 16 bits:

SF=8 bits, nº de líneas 28=256

SC=8 bits, nº de líneas 28=256, en total 512 líneas.

Dimensionamiento del bus de direcciones: Son los bits que tiene que tener el registro S para direccionar una memoria de X Kb.

X Kbyte−−−−−−−> N bits

2n=X Kbytes−−−−> log 2n=log 1024X N=(log 1024 X) / log 2

2.4.4 INTERCONEXION DE VARIOS CHIPS DE MEMORIA

Aumento del tamaño de palabra

Utilizando memorias con M bits en el bus de datos y N bits en el de direcciones queremos crear lo mismo pero con palabras de P bits.

Necesitaremos P/M chips.

Los buses de dirección y de control de todos los chips se conectan en paralelo y se dedican las líneas de datos (M bits) de cada memoria a una parte del bus de P datos del sistema.(Hoja adjunta)

Aumento de la capacidad de memoria

Para conseguir una memoria de X Kb a partir de circuitos de Y Kbytes, necesitamos X/Y chips.

Se conectan todos los buses de datos y de direcciones de todas las memorias entre sí.

Se interconectan en paralelo las líneas de control excepto CS ó equivalente.

Se crea un decodificador que tendrá al menos tantas salidas como circuitos de memoria se utilicen. Con estas salidas se controlarán los CS de cada memoria.(Hoja adjunta)

2.4.5 CRONOGRAMAS DE ACCESO A LA MEMORIA

1 no hay cambio alta

tiene impedancia

0 Rampa información

Cambio de información

Ciclo de lectura en memoria RAM

Intel 2114

En este tipo de memorias, el decodificador de selección trabaja permanentemente;

1.−En el bus de direcciones se introduce la dirección a buscar en memoria.

2.−Hay que decir a la memoria que se de por enterada (activar `CS')

3.−La palabra encontrada pasa a la salida del registro de memoria.

4.−Se desactiva CS

5.−Desaparecen los datos en el bus de ídem.

(Ver hoja adjunta de cronogramas)

Un tiempo crítico es el paso de ir del paso 1 al paso 3. Éste es el tiempo de acceso(TA)

que es el mínimo tiempo invertido desde que se introduce la dirección en el registro de direcciones hasta que aparece la palabra en la salida del registro de palabra.

Otro tiempo crítico es del paso 2 al paso 3(TCO=Chip Selection to Output Valid)

Entre TA y TCO determinan el tiempo de salida de la palabra.

Otro tiempo crítico es del paso 4 al 5. Desde que se desactiva CS hasta que se desactiva la palabra (TOTD=Output Triestate from Deselection)

Ciclo de escritura en memoria RAM

Pasos: (Ver hoja adjunta)

1.− Entra la dirección en el registro S.

2.− Se activa CS y WE.

3.− La palabra se introduce en M(se pone en el bus de datos) y entra en el registro.

4.− Se desactiva CS y/ó WE.

5.− Se elimina la palabra en el bus de datos(M).

Un tiempo crítico es Tw(Write Timer) que es el tiempo desde el paso 2 al paso 4. Es el tiempo mínimo que debe estar activos CS y WE.

Otro es el Tdw (Data to Write timer overlap) que es el tiempo transcurrido desde el paso 3 al 4. Es el tiempo

mínimo que debe permanecer la palabra M antes de desactivar CS ó WE.

Otro más es el Tdh que es el tiempo desde el paso 4 al 5 y que es el tiempo mínimo que la palabra que se ha introducido en M ha de permanecer desde que se desactiva CS y/ó WE, para asegurar que las puertas triestado deben de conducir.

Ciclo de lectura de ROM

Se diferencia de la RAM en que la dirección no necesita poder estar estable durante todo el tiempo de lectura.

Tcyc (Cyce Time): Ciclo de tiempo de CE. Es el tiempo mínimo necesario desde que comienza un ciclo CE hasta que puede comenzar el siguiente ciclo CE.

Tah (Address hold time): Tiempo necesario de permanencia de la dirección desde que se activa CE necesario para almacenar la dirección internamente.

Tas (Address to E setup time): CE puede activarse en el momento justo en el que aparece una dirección estable en S, no antes. Tiempo entre dirección y activación de CE (mínimo=0). Desde que se activa CE hasta que se obtiene la palabra. Sus tiempos son Tce y Toe.

Tce (CE to Output Delay): Es el más desfavorable. Es el que marca el tiempo transcurrido en CE y tenemos el dato y es el mínimo invertido en presentar la palabra desde que aparece CE.

Toe(OE to Output Delay): Es el menos desfavorable y es el que tarda las puertas triestado en transferir el contenido del registro M a la salida.

Ta (Address to output delay): Tiempo mínimo que debe haber transcurrido desde que tenemos la palabra estable hasta que tenemos salida.

Zonas de desactivación:

Tdf (OE to data float):Es el tiempo que tarda en dasaparecer los datos de la salida una vez desactivado OE.

Tcc(CE off time): Tiempo mínimo necesario para tener desactivado CE antes de volver a activarlo.

2.4.6 ESTADOS DE ESPERA

Es un acceso a memoria a grosso modo que consiste en tres pasos:

1.−Poner la dirección del Bus de direcciones.

2.−Activar la lectura.

3.−Lectura del dato.

Estas órdenes las gestiona la CPU.

Si el tiempo de duración a una operación es superior a un ciclo, se leerá alta impedancia, no el dato.

Supongamos una memoria con tiempo de acceso de 80 ns.

Caso 1: El reloj de la CPU a 10 Mhz.

¿Cuanto dura un ciclo de reloj?

1 ciclo= 1 / (10*106)=10−7 seg= 100 ns

1 ciclo de lectura=2 ciclos de reloj=>200 ns.

Ta(80 ns)<2 ciclos(200 ns)=>Funcionaría.

Caso 2: El reloj de la CPU a 33 Mhz.

¿Cuanto dura un ciclo de reloj?

1 ciclo= 1 / (33*106)=30 ns.

1 ciclo de lectura=2 ciclos de reloj=>60 ns.

Ta(80 ns)>2 ciclos(60 ns)=>No funcionaría.

Por estos errores se inventan los estados de espera, que se generan por una línea que se activa. Cuando la CPU tiene que esperar, ésta permanece en ciclos sin hacer nada.

Cada ciclo de reloj que la CPU permanece en espera se llama estado de espera(WAIT STATE)

Velocidad Reloj (Mhz.)	Tiempo Estado Espera(ns)
4,77	200
10	100
16	63
25	40
33	30
50	20
66	15
90	11

2.5 OPTIMIZACION DEL TIEMPO DE ACCESO A MEMORIA

2.5.1 MEMORIA INTERLEAVE(Entrelazado de memoria, interpaginada, intercalada)

Se basa en ejecutar simultáneamente diferentes fases de diferentes instrucciones. La memoria se divide en varios bloques, cada bloque tiene su bus de direcciones y de datos, y pueden trabajar simultáneamente. La técnica consiste en que el micro puede pedir anticipadamente datos a cada bloque y con esto lo que consigue es tener los datos leídos antes de que los necesite leer.

Para hacer la mejora de tiempos:

−.− El primer acceso a memoria de la CPU tiene estado de espera.

−.− La CPU ya tiene la siguiente instrucción.

Si el dato que necesita está en el siguiente bloque, hace inmediatamente la petición de lectura antes de recibir

el dato del bloque que ésta esperando.

En general el aumento de prestaciones viene a ser un 20%. Si el programa es muy secuencial, los datos se darán alternados.

Con un SO multitarea puede pasar cualquier cosa. Cuanto más secuencial sea, mayor es el rendimiento.

2.5.2 MEMORIA CACHÉ

Se necesita de un 386 en adelante. Utiliza DRAM(60 ns en tiempo de acceso)

Consiste en tener una pequeña parte de DRAM con 10 ns. Se quiere tener en esa DRAM cualquier dato que se va a requerir. Pero no siempre se consigue, porque hay que predecir cual son los siguientes datos que la CPU necesitará para grabarlos en caché previamente. No es posible saber con certeza que datos va a querer la CPU pero se siguen las siguientes pautas:

1.−Probable secuencialidad, si se accede a una posición de memoria determinada, es muy probable que los siguientes se realicen a las siguientes posiciones.

2.−Existen datos de acceso muy frecuentes(bucles, variables, pilas,...).

Si un dato al que se desea acceder está en la caché se le llama acierto de caché y si no está se le llama fallo de caché.

La caché pretende agilizar los tiempos, por lo que se accede a la DRAM y a la caché en ambos tiempos.

La orden de magnitud de las tasas de aciertos suele ser de 80..95%.

Parámetros que influyen en los rendimientos de la caché:

1.−El tamaño de la memoria principal.

2.−El tamaño de la caché.

3.−El tamaño de loa bloques de memoria que se copian en caché.

Cuánto mayor sea la caché, mayor será la tasa de aciertos.

En el tamaño de línea(bloque):

− Sólo se debe a la presunción de secuencialidad.

− Es del orden de 2, 4, 8, 16 bytes.

El valor es pequeño por tres motivos:

*Cuanto menor sea el bloque, más nº de bloques tendrá la caché.

*Si el bloque es pequeño, tarda menos en transferirlo.

*Cuanto mayor sea el bloque, menor probabilidad hay de que sus últimos bytes se utilizan por estar alejada del byte pedido.

Tipo de organización de datos:

Hay que diferenciar unos parámetros que influyen;

− Tipo de criterio sobre que bloque eliminar para dejar espacio a uno nuevo. Se hace eliminando el bloque que menor tiempo ha sido accedido.

− La forma de almacenar a los bloques en caché; Se tiene una tabla dónde se almacenan los identificadores de bloque, consultando a la tabla sabe la caché si está o no está el dato, a esta tabla se le llama memoria de etiquetas. Cuando se halla el tamaño de la caché no entra el tamaño de memoria de etiquetas(sólo la de datos). Según se reorganice la memoria de etiquetas se conocen tres tipos de caché:

2.5.2.1 Caché asociativa.

Es la caché completamente asociativa y la tabla contiene la dirección completa de todos los bloques. Cuando se realiza un acceso, es comparar el valor de la dirección a buscar con los que tienen en caché.

INCONVENIENTES: Hace falta la memoria CAM que es cara, y de gran tamaño.

Este tipo de RAM necesita tener un circuito de control para decidir que bloque eliminar para dejar sitio a un nuevo bloque.(tantas comparaciones como líneas).

2.5.2.2 Caché directamente mapeada.

Con una sola comparación se sabe si el bloque está ó no en caché.

Elegido un bloque sólo existe un sitio determinado dónde colocarlo dentro de la caché.

El identificador del bloque se divide en dos partes: Indice y etiqueta.

El tamaño del índice se corresponde con el nº de líneas de la caché. La etiqueta tiene el resto (la palabra de direccionamiento del bloque menos el índice). Lo que se hace para comprobar si está o no una dirección, se quita los ocho bits menos significativos de la dirección de bloque y con el resto se compara en la posición que indica.

INCONVENIENTES: Si se accede frecuentemente a posiciones iguales que tengan sus ocho bits menos significativos iguales , entonces se está constantemente cambiando al contenido de la caché y constantemente a la RAM y sólo uno de ellos puede estar en un momento determinado en la caché.

VENTAJAS: No es necesario ningún circuito de control para eliminar un bloque.

(una sola comparación)

2.5.2.3 Asociativa por conjuntos.

Se divide en varios bloques toda la caché(2 ó 4 habitualmente).

Cada uno de estos subgrupos se maneja como una caché directamente mapeada.

Hay que hacer dos comparaciones porque hay dos direcciones iguales en cada subbloque

Este tipo es el más utilizado porque es el que mejor rendimiento tiene. Se pone un biestable en cada una de las

dos direcciones que indica cuál ha sido el último bloque accedido

2.5.3 MEMORIA SHADOW RAM (sombra)

Mantiene en memoria RAM partes de la ROM con lo que se mejoran los tiempos de acceso. Ej: Rutinas de acceso a unidades de disco, al teclado, a la tarjeta de vídeo...)

2.5.4 EDORAM (Extended Data Out RAM)

RAM de salida de datos extendida. Salió a mediados del 95 y el primer acceso a memoria lo hace con estados de espera iguales a cualquier otra RAM, pero los siguientes, si son secuenciales, no tienen estados de espera.

TEMA 3. LA UNIDAD ARITMETICO LOGICA (A.L.U.)

3.1 ALU INTEGRADA Resultado

Salidas de estado Entradas de control

Dato 1 Dato 2

La ALU es un circuito combinacional capaz de realizar operaciones aritméticas y lógicas.

Por las entradas de datos entran los operandos (números con los que operan).

El tamaño de la ALU no depende del bus de datos y no hay un registro en la ALU para almacenar la salida, el resultado son siempre líneas de datos.

Las entradas de control indican el tipo de operación a realizar, dependiendo del número de líneas de control habrá un número de operaciones.

Las salidas de estado indican ciertas características de las operaciones que se han dado:

*Carry *Overflow *Signo *Si el resultado es 0.

El contenido de las salidas de estado se almacena en unos biestables llamados **Flags.**

Externo a la ALU suele haber un registro llamado acumulador que es capaz de almacenar una palabra.

3.2 TIPOS DE DATOS NUMERICOS

3.2.1 ENTEROS SIN SIGNO

Sólo se pueden representar datos positivos y tienen una codificación binaria normal.

Suponiendo que el dato tiene n bits; ninguno de estos es para signo.

El rango de valores que puede tener es [0..2n−1], por lo tanto 2n.

n depende de la ALU.

3.2.2 ENTEROS CON SIGNO

Son números enteros positivos y negativos, se codifican en CA2. Un dato de n bits tiene un bit de signo. El rango es [−2n−1..2n−1−1], es decir 2n valores

3.2.3 DATOS REALES

Son valores enteros negativos y positivos y números no enteros(reales). Se codifican en coma flotante.

− Existen ALU's para enteros y para reales.

3.3 FLAGS Y SU SIGNIFICADO

Los Flags son las salidas de estado de la ALU, se almacenan en biestables para luego ver que ha pasado en una operación. Se utilizan para provocar bifurcaciones en futuras instrucciones y para poder operar con palabras de tamaño mayor a 1 byte.

Todos los Flags se guardan juntos en un registro y cada bit de ese registro tiene un significado concreto:

3.3.1 CARRY(FC)

Se corresponde a la salida Cn de un sumador. En enteros con signo indica desbordamiento en la operación. En una suma si hay desbordamiento Cn=1 y en una resta si hay desbordamiento, Cn=0(Carry Borrow).

En desplazamiento a derecha ó izquierda(Shift), el bit que sale va al Carry. En un Shift a derecha, en el Carry hay el resto y en un Shift a izquierdas tenemos el desbordamiento.

3.3.2 OVERFLOW(F0)

Es un XOR entre Cn y Cn−1.

Indica desbordamiento en operaciones aritméticas con enteros con signo.

1 indica desbordamiento.

3.3.3 FLAGS DE SIGNO (FS)

Es una copia del bit de más significativo(el de signo) del resultado.

En enteros con signo, FS=1 ó FS=0, en enteros sin signo no tiene significado.

3.3.4 ZERO (FZ)

Indica si el resultado de las operaciones es 0 ó distinto de cero.

FZ=0, el resultado ha sido 0.

3.4 TIPOS DE OPERACIONES

3.4.1 ARITMETICAS

3.4.1.1 *Suma con Carry*

A+B+FC=C(rgtro destino)

Realiza la suma de dos operandos A y B y si el carry es 1 se incrementa en 1, sino nada.

3.4.1.2 *Resta con Borrow*

A−B−FB=C

Si el borrow es 1, decrementa.

3.4.1.3 *Comparación*

Se realiza mediante una resta. Resta los operandos A y B, actualiza los Flags pero no se almacena esa operación.

3.4.1.4 *Cambio de signo*

Con signo: 0−A−>A

Si A es entero sin signo: 2n−A−>A

En sumas y restas sin Carry ni sin Borrow influyen.

3.4.2 TIPOS DE DESPLAZAMIENTOS Y ROTACIONES

3.4.2.1 Desplazamientos

IZQUIERDA A<A

C

0 (lo mismo que x2)

DERECHA:

*Lógico: A>>A

0 Dato Resto No existe desbordamiento

*Aritmético: A>A

Bn Resto

bit de mayor peso

3.4.2.2 Rotaciones

*Normal

Carry

*Rotación a través del carry:

Carry

3.5 LENGUAJE SIMBOLICO

En fotocopias.

3.6 OPERACIONES CON NUMEROS DE VARIOS BYTES

3.6.1 SUMA (Con ó sin signo):

a) Se suma sin carry los bytes menos significativos

b) Con los significativos (de derecha a izquierda) suma con carry byte a byte.

Desbordamiento:

Enteros sin signo(ESS): Se activa Fc en la suma del último byte.

Enteros con signo(ECS): Se activa Fo en la suma del último byte.

3.6.2 RESTA (Con ó sin signo):

a) Suma sin borrow de los bytes de menor peso significativos.

b) Con los significativos (de derecha a izquierda) con borrow byte a byte.

Desbordamiento:

ESS: Se activa Fb en la suma del último byte.

ECS: Se activa Fo en la suma del último byte.

3.6.3 MULTIPLICACION x2 (Con y sin signo):

a) Desplazamiento hacia la izquierda del byte menos significativo.

Si la máquina no tiene esto, se puede hacer con rotación a través del carry teniéndolo primero a 0.

b) Después hacia la derecha, byte a byte, rotación izquierda a través del carry.

Desbordamiento:

ESS: Se activa Fc en el último byte.

ECS: Se activa Fo en el último byte.

Si además resultara que el desplazamiento con rotación no actualizara los flags entonces se compara Fc con Fs. (Bn <> Bn−1)

3.6.4 DIVISION x2 (Con ó sin signo):

a) ESS: Al byte más significativo hay que hacerle un Shift a la derecha.

ECS: Al byte más significativo, hacer un desplazamiento aritmético hacia la derecha, en caso de que no hay

desplazamiento aritmético, habrá que hacer una rotación a derecha a través de carry, pero poner previamente el carry a 0 en positivos ó a 1 en negativos.

b) De más a menos significativos, de izquierda a derecha, hacer rotación a derecha a través de carry. No hay desbordamiento pero el Fc queda con el Resto.

3.7 APLICACIONES DE LAS OPERACIONES LOGICAS

3.7.1 MANIPULACION DE BITS

palabra: bn bn−1 bn−2 ... b0

a) Forzar 0's con AND:

bi ^ 0 −> 0

bi ^ 1 −> bi

b) Forzar 1 con OR:

bi v 0 −>bi

bi v 1 −> 1

c)Complementario con XOR:

bi " 1 −> bi

bi " 0 −> bi

3.7.2 COMPROBACION DE QUE UN OPERANDO DE VARIOS BYTES DA 0.

A v B −> 0 si A=0 y B=0

OR

OR

OR

Fz

Byte 0 OR Byte 1 −−>A

A v Byte 2 −−> A

A v Byte 3 −−> A :: Fz=1 si todos son 0.

3.7.3 ACTIVACION DE FLAGS

bi ^ bi −> bi

bi v bi −> bi

3.7.4 COLOCAR UN CERO EN EL ACUMULADOR ACTIVANDO FZ .

bi " bi −> 0

3.7.5 COMPROBACION DE IGUALDAD DE REGISTROS

Es un caso particular del apartado 3.7.4

A " B −−> A si FZ=1, son iguales

bi " bi −−> 0 si FZ=0, son diferentes

3.8 USO DE LOS FLAGS EN COMPARACIONES

a) En enteros sin signo sólo nos interesan FC y FZ

A−B

FZ	FC	Significado
0	0	A>B
0	1	A<B
1	0	A=B
1	1	No existe

b) En enteros con signo nos interesan FZ , FS , Fo

A−B

FZ	FS	FO	Significado
0	0	0	A>B
0	0	1	A<B
0	1	0	A<B
0	1	1	A>B
1	0	0	A=B
No existen más combinaciones con FZ=1			

Tabla Resumen

Conclusión	SIN signo	CON signo
A>B	FC=FZ=0	FZ=0 Y FS=FO
A=B	FZ=1	FZ=1
A<B	FC=1	FS<>FO
A>=B	FC=0	FS=FO
A<>B	FZ=0	FZ=0
A<=B	FC<>FZ	FZ=1 O FS<>FO

TEMA 4. LA UNIDAD DE CONTROL

4.1 INTRODUCCION

La unidad de control es el mecanismo encargado de gobernar el funcionamiento del ordenador. Funcionalmente recibe la información almacenada, la decodifica, genera las señales de control (microórdenes) necesarias para realizar las transferencias, controles, etc. Las características de la UC son las mismas que las de la CPU. El Indalo 1.0 es una pequeña CPU.

4.2 REGISTROS

Se utilizan para almacenar datos de forma temporal:

Registros de propósito general: Almacena indistintamente datos y/ó direcciones y no siempre el mismo tipo de dato y/ó dirección. Su longitud, por lo tanto, será variable y dependerá del uso que se le dé en cada instante.

Registros de propósito específico: Almacenan siempre el mismo tipo de dirección ó dato. Es de longitud fija.

En el Indalo 1.0 sólo son de propósito específico y son de 8 bits de longitud.

Las salidas de los registros se conectan a la UC por medio de puertas triestado (buffer triestado).

4.2.1 REGISTRO DE INSTRUCCION (IR)

En cada momento contiene la instrucción en curso, igualmente contiene el dato con la dirección asociada a esa instrucción.

Instrucción=Código de operación + DIR (dirección ó dato)

Se puede decir que el IR está compuesto de varios registros, en Indalo 1.0 el CO es un registro y el DIR son 2.

El CO indica el tipo de operación que se va a realizar y el operando indica sobre qué va a actuar esa operación. Va conectado al decodificador que a su vez va al secuenciador. Su longitud determina el número de operaciones que se pueden realizar y tiene que ser del tamaño de la palabra de memoria en la que se almacenan los códigos de operaciones.

El registro DIR es el resto de los bits del IR y tiene el tamaño del bus de direcciones.

En el caso de Indalo 1.0, el CO es 1 byte y el DIR, 2 bytes que se dividen en DIRH y DIRL (alta y baja).

4.2.2 CONTADOR DE PROGRAMA

Contiene la dirección de la próxima instrucción a realizar. La longitud del contador de programa será la del bus de direcciones(16 bits en Indalo 1.0). Si se activa el RESET, el contador de programa accede a una posición ROM (0FFFDH en el Indalo 1.0)

4.2.3 REGISTRO ACUMULADOR

Se le llama A y en el se guarda el resultado de la última operación que se procesa en la ALU.

4.3 SECUENCIADOR CENTRAL

Gobierna todos los elementos de la CPU. Funciona a través de unas microórdenes que las genera en función del estado de sus entradas y el estado de señales de control que le vienen de otros circuitos como pueden ser demandas de interrupción, estado de memoria, estado de la ALU, etc. y en función de cómo esté implementado físicamente el secuenciador podemos tener una UC cableada ó una UC programada.

El hecho de que la UC sea cableada quiere decir que el secuenciador está hecho por hardware, por lo que sólo puede realizar un sólo algoritmo. El diseño es muy rígido pero más rápido. Si es programada, el hacer un cambio de órdenes es muy sencillo, pero es más lenta. Lleva una pequeña memoria donde se colocan las órdenes. Se pueden añadir y modificar instrucciones modificando el microprograma. En función del secuenciamiento de las órdenes, se dividen en:

SINCRONOS: La UC conoce los tiempos de respuesta de los circuitos. Al lanzar una orden sabe cuanto tiempo tiene que dejar hasta ejecutar la siguiente orden.

ASINCRONO: Tiene que ir comprobando la ejecución de cada tarea para ejecutar la siguiente. Son más rápidos porque no utilizan ciclos enteros de tiempo.

4.3.1 DECODIFICADOR DE INSTRUCCIONES

Transforma los bits de CO en diferentes líneas del decodificador y activa la línea, indica al secuenciador que operación hay en CO. Indalo 1.0 sólo tiene 4 instrucciones (puede tener hasta 28).

4.3.2 MICROORDENES / MICROINSTRUCCIONES

Activar ó desactivar una orden es activar ó desactivar una línea. Las microórdenes se escriben en minúscula, mientras que las microinstrucciones se hacen en mayúsculas.

MICROORDENES:

De nivel: Actúan entre dos flancos (1 ciclo ó varios).Ej.: Poner la información en un bus.

Impulsionales: Actúan sólo durante el flanco de subida ó de bajada. Ej.: Leer la información de un bus en un momento.

de Nivel circuito Impulsional orden impulsional en flanco de bajada.

orden de nivel activa a nivel bajo.

DE NIVEL (pág. 4 fotocopias)

mem: activa la línea de acceso a memoria.

saldir: activa las puertas triestado del registro que lo conectan eléctricamente.

salpc: lo mismo, pero con el registro PC.

sala: lo mismo, pero con el Acumulador.

alu0: controla la ALU:

− Si alu0=0, transfiere ENT2 a la salida de la ALU.

− Si alu0=1, transfiere ENT1+ENT2 a la salida de la ALU.

reg1: − Si reg1=1, PC se pone en modo contador (CNT).

− Si reg1=0, PC se pone en modo Load (LD).

reset: Inicializa el secuenciador y el PC.

IMPULSIONALES

CK***: Reloj de entrada a ***.

rd y wr se activan en flanco de bajada (¡¡¡ no están negadas!!!)

4.3.3 CICLOS DE MAQUINA (Fotocopias pág. 5)

El ciclo de máquina trata del tiempo que tarda en realizar una transferencia a través de un bus externo.

En Indalo 1.0: Lectura de memoria

− Lectura de dato CO − 1 ciclo

− Lectura de instrucción DIRH − 1 ciclo Fetch

DIRL − 1 ciclo

Un ciclo fetch parte de la dirección de PC, una vez que acaba hay que incrementar PC, por esto, un ciclo fetch necesita más tiempo.

4.4 BUSES

Bus de datos: Dbus

Bus de direcciones: Abus

Bus de control: 15 líneas + 1 de alimentación.

Las 15 líneas son 1 CK + 14 Microórdenes.

4.5 INSTRUCCIONES

Se pueden indicar 28 instrucciones de las que Indalo 1.0 sólo trabaja con 4.

DIRH + DIRL nos indica el operando, porque lo sea en sí mismo ó porque nos indique el lugar dónde se encuentra.

CO, DIRH y DIRL son 3 bytes consecutivos en memoria, sus posiciones serán respectivamente n, n+1 y n+2.

órdenes=líneas activas ó desactivas.

A+B−−>A instrucciones (transferencia entre regisgtros)

4.5.2 SINTAXIS

Mnemónicos(ensamblador)−−>mayúsculas

Mnemónico[operando1[operando2]]

operando1: destino operando2: fuente

Ej.: ADD A,operando2 => A+operando2−−>A

Suma: ADD A,(address): Aritmética Direccionamiento Directo

MOV A,(address): Cargar el acumulador con el dato de la posición indicada.

MOV (address),A: Cargar el contenido del acumulador en la posición indicada.

JMP address: Salto incondicional / Direccionamiento absoluto, pone la dirección en PC.

4.6 EJECUCION DE LA INSTRUCCION

1.− PC vuelca su contenido en Abus.

2.− Generar señal de lectura en memoria.

3.− El dato pasa al Dbus. ciclo fetch

4.− Activar lectura de CO.

5.− El decodificador la decodifica.

6.− Activar CNT del PC

7.− En PC aparece la posición n+1.

8.− PC vuelca en Abus la posición n+1.

9.− Pasos 2 y 3.

10.− Activar lectura de DIRL.

4.6.1 FASES DE EJECUCION DE UNA INSTRUCCION

Fase 1: Búsqueda y análisis de la instrucción

PC−−>Abus

(Abus)−−>Dbus

Dbus−−>CO

PC+1−−>PC

Fase 2: Ejecución de la instrucción

Página 6 de fotocopias.

En JMP se aplica un `reset' al secuenciador para comenzar una nueva instrucción.

TEMA 5. CPU BASICA

5.1 CARENCIAS DE INDALO 1.0

Tiene pocos registros, no tiene tipos de direccionamiento, la ALU tiene pocas operaciones de proceso. No tiene Flags y no se pueden hacer saltos condicionales. Las instrucciones son muy reducidas.

5.2 REGISTROS AÑADIDOS

C y B de 8 bits que no están conectados a la ALU como el A. Estos registros no se pueden utilizar en operaciones lógicas ni aritméticas, sólo en transferencias(MOV), incremento ó decremento. Pueden trabajar como uno de 16 bits, entonces se llama BC.

X de 16 bits, que está destinado a direcciones de memoria, en particular a direccionamientos indexados, se puede incrementar ó decrementar su contenido.

Tipos de operandos

Los operandos pueden ser de 8 ó de 16 bits:

op8 Operando de 8 bits que puede ser:

a)Registro de 8 bits Ej.: **A**+B−>A

reg8: A, B ó C

b)Dato contenido en memoria Ej.: A+(38H)−>A

dat8: Dato en memoria a continuación de su CO.

c)Valor inmediato de 8 bits(en Hexadecimal) EJ: A+38H−>A

mem8: Dato en memoria de 8 bits.

− Direccionamiento por valor (address)

− Direccionamiento indirecto por registro (BC) ó (x+rel8)−−>indexado

En el caso de por valor, la dirección está contenida en los 2 bytes a continuación del CO(como en Indalo 1.0) `CO+2bytes'

rel8 Desplazamiento relativo de 8 bits. Entero con signo (de −128 a 127). Es un byte que aparece después del CO `CO+1 byte'

op16 Operando de 16 bits.

*Dato inmediato en registro de 16 bits (BC ó X) **reg16**

*Dato inmediato en los 2 bytes siguientes al CO. **dat16**

5.3 TIPOS DE DIRECCIONAMIENTO

[*]Direccionamiento para buscar operandos

***Inmediato:** En la instrucción incluye el valor del operando. Ej.: 45H−−>A

***Directo:** La instrucción expresa dónde está el contenido del operando.

Ej.: (4507)−−>A ó B−−>A

***Indirecto:** La instrucción expresa el lugar dónde está contenida la dirección del

operando de memoria. Ej.: (X)−−>A

[*]Direccionamiento implícito ó inherente: La instrucción no explícita el operando porque el código de operación(CO) lo tiene implícita.

INDALO II utiliza los siguientes direccionamientos:

Inmediato: dat8

Directo por registro: A, B ó C

Directo por valor: (address)

Indirecto por registro: (BC)

Indirecto por registro indexado: (X+rel8)

La dirección que contiene X+el valor entero con signo de 8 bits que indica rel8, será la dirección dónde se encuentra el operando.

[*]Direccionamiento en instrucciones de ruptura de secuencia.

Ruptura de secuencia son aquellas instrucciones que no siguen un secuenciamiento (saltos, bucles y bifurcaciones).

***Absolutas:** La instrucción expresa la dirección dónde se debe de saltar. CO+dir.

***Relativo a contador de programa:** La instrucción expresa una magnitud de un byte (que se considera entero con signo) que debe sumarse al registro PC.

CO+1 byte−−>entero con signo=>Dir. salto=PC+byte

Cuándo se ejecuta la instrucción de salto ya está el registro PC en la dirección siguiente, lo que habrá que tener en cuenta.

***Indirecto por registro:** La instrucción expresa el registro en el cual se encuentra la dirección a cargar en PC

para que salte. Si en ese registro hay una dirección, el registro tiene que ser de 16 bits, no vale uno de 8. Ej.: Salta a (BC)

*__Indirecto por valor:__ La instrucción expresa una dirección de memoria que contiene, junto con la siguiente, la dirección a la que hay que saltar.

En INDALO II tenemos:

Absoluto: El direccionamiento es una dirección de 16 bits.

Indirecto por registro: El direccionamiento es un registro de 16 bits.

Relativo condicional: Es del tipo relativo al contador de programa, se utiliza en saltos condicionales. El direccionamiento es un dato de 16 bits, ya que es más cómodo trabajar con la dirección completa, aunque la máquina trabaje con desplazamientos.

5.4 OPERACIONES AÑADIDAS A LA A.L.U.

Tenemos cuatro líneas de microórdenes (alu0, alu1, alu2 y alu3) por lo que tenemos 24 operaciones posibles, que son:

0000 Ent2−−>Result La entrada 2 de la ALU pasa a result.

0001 ADD Suma la entrada1 y la entrada2 sin Carry.

0010 ADC Suma la entrada1 y la entrada2 con Carry.

0011 AND And lógico entre entrada1 y entrada2.

0100 Ent2−− Decrementa entrada2 y lo pone en Result.

0101 Ent2++ Incrementa entrada2 y lo pone en Result.

0110 NEG Calcula el CA2 del acumulador y lo pone de nuevo en el acumulador.

0111 NOT Calcula el CA1 del acumulador y lo pone de nuevo en el acumulador.

1000 OR OR lógico entre el acumulador y entrada2, y de nuevo al acumulador.

1001 RCL Rotación del acumulador a izquierdas a través del carry. A<C< A

1010 RCR Rotación del acumulador a derechas a través del carry. A>C> A

1011 SBB Resta al acumulador la entrada2 con Carry A−[Ent2+Fc]−−>A

1100 SHL Shift (desplazamiento) lógico a izquierda del acumulador.

1101 SHR Shift lógico a derechas del acumulador >>

1110 SUB Resta del acumulador la entrada2 sin tener en cuenta el Carry.

1111 XOR OR exclusivo del acumulador y entrada.

5.5 FLAGS DE INDALO II

El INDALO I no tenía ningún Flag, mientras que INDALO II tiene 5.

Los Flags dependen de las operaciones de la ALU.

FC Flag de Carry:

En suma es el acarreo

En Resta o comparación es el Borrow.

En desplazamiento ó rotación es el Bit saliente.

En una operación lógica es siempre=0.

FZ Flag de cero: Se pone a 1 si el resultado (result) es 0.

FO Flag de Overflow:

En operaciones aritméticas hace un XOR de Cn y Cn−1, es decir, el acarreo de los 2 bytes más significativos. Sólo actúa en enteros con signo.

En desplazamientos y rotaciones se pone a 1 si el bit de mayor peso cambia de signo.

En operaciones lógicas siempre es 0.

FS Flag de signo: Es el bit más significativo de Result.

FP Flag de paridad:

0 si en Result hay un número impar de 1's.

1 si en Resulr hay un número par de 1's.

5.6 SET DE INSTRUCCIONES DE INDALO II.

5.6.1 INSTRUCCIONES DE TRANSFERENCIA

Todas utilizan en mnemotécnico MOV, no modifican Flags. Dos grupos:

*Transferencias de 8 bits:

MOV reg8,op8

El destino es un registro de 8 bits (A, B ó C)

La fuente es cualquier operando de 8 bits.

MOV mem8,reg8

El destino es una posición de memoria.

La fuente sólo puede ser un registro de 8 bits (A, B ó C)

*Transferencias de 16 bits:

MOV reg16,op16

El destino son registros de 16 bits.

La fuente puede ser cualquier operando de 16 bits.

5.6.2 INSTRUCCIONES ARITMETICAS

Todas afectan a los Flags. El único destino posibles es el acumulador y como fuente hay siempre dos posibles operandos: Operando1 que es siempre el acumulador y Operando2 que en caso de existir puede ser cualquier op8.

ADC A,op8 Suma con acarreo.

ADD A,op8 Suma sin acarreo.

SBB A,op8 Resta con Borrow. A−(op8−FC)−−>A

SUB A,op8 Resta sin Borrow.

CMP A,op8 Es una resta sin resultado alguno, sólo actualiza Flags.

NEG A Cambio de signo.

5.6.3 INSTRUCCIONES LOGICAS

Todas afectan a los Flags. El Fo y el Fc se pone a 0 al realizar cualquier instrucción lógica. El destino es siempre el acumulador. La fuente es cualquier op8, caso de existir.

AND A,op8

OR A,op8

XOR A,op8

NOT A

5.6.4 INCREMENTOS y DECREMENTOS

Hay dos tipos de situaciones:

*Operandos de 8 bits:

Necesitan que se actualicen flags, ya que es un dato, por lo que la operación se realiza a través de la ALU. La fuente y el destino son el mismo registro. Son las únicas operaciones que realiza la ALU en las que el origen y el destino son los mismos y no es el acumulador, sino cualquier otro registro: **INC reg8 DEC reg8**

*Operandos de 16 bits:

No es necesario hacer la operación en la ALU, ya que al ser una dirección no hay que actualizar flags. La operación se realiza en el propio registro, activando la entrada de contador UP/DOWN. Supone un ahorro importante de tiempo respecto a hacerlo a través de la ALU.

INC reg16 DEC reg16

5.6.5 SHIFT y ROTACIONES

Afectan a los flags, y el de carry toma el valor de bit saliente y el de overflow se pone activo si cambia el bit de signo y se pone a 0 si hay un desplazamiento a derechas.

Rotación:

RCR A Derecha A>C> A

RCL A Izquierda A<C< A

Desplazamiento:

SHL A Izquierda A< A

SHR A Derecha A> A

5.6.6 INSTRUCCINES DE MANIPULACION DE FLAGS

Con INDALO II sólo se puede modificar el flag de carry.

STC Pone Fc a 1

CLC Pone Fc a 0

5.6.7 INSTRUCCIONES DE SALTO

a)Incondicional

Todas tienen un mnemónico general que es **JMP** y se ejecutan siempre que aparecen. No modifican flags. Los direccionamientos que utilizan son:

JMP reg16 Indirecto por registro (BC ó X)

JMP address Absoluto (address)

b)Condicional

Se ejecuta el salto si se cumple una determinada condición, que es el estado de un flag determinado. Existen dos instrucciones de salto con cada flag. El mnemónico es una J seguida de una N si es no activo y la inicial del flag

JC address Si Fc=1, PC+rel8−−−>PC

JNC address

JO address

JNO address

JP address

JNP address

JS address

JNS address

JZ address

JNS address Si Fs=0, PC+rel8−−>PC

Ejecución de un salto condicional:

Se lee el CO(1 ciclo fetch), se comprueba si el flag vale lo que dice el CO; Si no se cumple, directamente incrementa PC sin leer el byte que viene a continuación(5 ciclos).

Si se cumple la condición, habrá que leer rel8(1 ciclo fetch). (10 ciclos).

5.6.8 INSTRUCCIONES DE CONTROL

NOP No operar. Es esperar, ya que dura 4 ciclos de reloj(1 ciclo fetch).

5.7 LENGUAJE ENSAMBLADOR

El ensamblador es el mnemónico y el programa que lo pasa a código máquina es el *programa ensamblador*.

Sintaxis:

*En cada línea hay una instrucción.

*Cada línea se compone de 4 campos:

**Campo de etiquetas(opcional). Una etiqueta es un máximo de 8 caracteres alfanuméricos que tiene que empezar por letra y no puede llevar espacios ni caracteres especiales ni de puntuación. Después de ella hay que poner (:).

**Mnemónico de la instrucción.

**Campo de operandos (puede existir ó no).

**Campo de comentarios(opcional). Antes de él tiene que haber (;).

*Se pueden poner líneas vacías.

*Existen pseudocódigos y directivos. Son palabras utilizadas en ensamblador como CO pero no son instrucciones de la CPU.

PSEUDOCODIGOS: Mnemónicos que pertenecen al ensamblador, no a la CPU y dependen del SO con el que se trabaje. En general son llamadas a subrutinas del SO. Cuándo el traductor lee un pseudocódigo, genera el código máquina equivalente.

DIRECTIVOS: Son también órdenes que entiende el ensamblador, pero que no generan código máquina, son órdenes al ensamblador.

DB (Define Bytes): Indica al ensamblador que debe cargar una serie de datos en bytes a partir de la dirección en la que se encuentra. Ej.: DATOS: DB 7,18,89

Está diciendo que abra 3 posiciones de memoria y que en la primera cargue un 7, en la segunda un 18 y en la tercera un 89. La etiqueta DATOS apuntará a la primera.

En ensamblador se permite la definición de etiquetas como variables y las referencias al contenido de una variable se utilizan con paréntesis. MOV A,(DATO)

DW (Define Words): Es lo mismo que DB pero los datos son palabras de 2 bytes que se pueden interpretar cómo alta y baja ó al revés. La dirección más baja va a ser el byte más significativo. Ej.: DATO: DW 7,1894,5,1200

Deja hueco para 8 bytes, 2 para cada palabra, primero se guarda la parte baja y después la alta. Así en la primera posición tendremos un 07 y en la segunda un 00, en la tercera un 94 y en la cuarta un 18,etc.

EQU : Asigna valor a una etiqueta. Ej.: PUERTO: EQU 32

Después de ejecutarse, cada vez que aparezca PUERTO, valdrá 32.

ORG : Va seguido de un dato de 16 bits entero sin signo, es decir, una dirección. Sirve para indicar al ensamblador a partir de qué dirección se carga el programa.

Si un programa se calcula con direcciones absolutas, obliga a que ese programa se cargue siempre en la misma posición de memoria. Para solucionar este problema se referencian todas las direcciones absolutas a la dirección de comienzo del programa. El ensamblador transforma todas en relativas a esa posición.

TEMA 6. CPU AVANZADA. INDALO III

6.1 CARENCIAS DE INDALO II

No hay facilidad para el manejo de subrutinas, no permite interrupciones ni DMA ni el manejo directo de E/S.

6.2 HARDWARE AÑADIDO

Registro **SP** (Stack pointer) que es el puntero de pila.

Un multiplexor situado bajo el PC que tiene dos bases en su entrada y sirve para poder meter tanto direcciones como datos.

Otro multiplexor debajo del DIR para hacer desplazamiento de esta a la izquierda para multiplicarlo por dos.

En el bus de control hay más líneas y microórdenes nuevas.

NMI: No Mask Interruption (Interrupción no enmascarable)

HOLD: Petición de buses por parte del DMA.

HOLDA: Acknowledge (Aceptación de la entrega de buses al DMA).

INT: Entrada de interrupción al microprocesador.

INTA: Acknowledge (Aceptación de la interrupción).

IO: Acceso de E/S.

Se añaden 8 puertas AND en los bits más significativos del Abus. Cuándo mem=0 se direcciona otra cosa que no es memoria.

Se añaden puertas triestado en las últimas líneas del bus de control.

Aparece el flag I, que es el flag de interrupción. Los flags están en un registro que se llama F 0 0 C Z O S P I

6.3 MANEJO DE SUBRUTINAS: LA PILA

Cuando termina de ejecutarse la subrutina se vuelve a la dirección siguiente a la que provocó la llamada. El SP apunta a la última dirección de la pila:

CALL address Llamada a subrutina. El contenido de PC se almacena en la pila, en la dirección apuntada por SP(que son dos bytes).

RET Retorno de subrutina. Cuándo nos encontramos en una subrutina con ésta orden, se leen los dos últimos bytes de la pila, y los carga en PC, se incrementa en dos posiciones SP.

La pila se puede desbordar, para evitarlo Indalo III parte de la última posición de memoria y crece hacia direcciones bajas.

6.4 OTRAS INSRUCCIONES ASOCIADAS A LA PILA

*Nos sirve también para el salto de interrupciones, almacenando direcciones.

*Almacena datos que pueden ser:

− Registros, mientras se realiza una tarea, por ejemplo, los Flags.

− Almacenar parámetros de la subrutina (no lo tiene Indalo III).

Estas operaciones se realizan mediante las instrucciones:

PUSH: Permite introducir direcciones ó datos de 16 bits en la pila, actualizando el valor de SP.

POP: Permite recuperar los valores almacenados de 16 en 16 bits.

PUSH BC Guarda en la pila el contenido del registro BC.

PUSH X Guarda en la pila el contenido del registro X.

PUSH AF Guarda en la pila el contenido del acumulador y los Flags.

POP AF Guarda en el acumulador y en los Flags el contenido de la pila.

POP X Guarda en X el contenido de la pila.

POP BC Guarda en BC el contenido de la pila.

Se recuperan en sentido contrario al que se guardan.

Al regresar de una interrupción, todo el contenido tiene que estar igual que antes de generarse dicha interrupción. Se guardan BC, X, A, F en la subrutina de atención a la interrupción.

6.5 INTERRUPCIONES

Indalo III utiliza interrupciones vectorizadas por dirección. El micro tiene una patilla (INT) por la que viene una señal de interrupción de algún interface, unidas todas por una puerta OR. INTA es la línea de reconocimiento de la interrupción, cuya señal se manda a todos los interface, pero el que la ha solicitado la reconoce como suya; Entonces genera un byte que lo pone en el bus de datos y que se llama *vector de interrupción* con el cuál se genera una dirección (dirección1), a la cuál se va y se leen dos bytes que dan la dirección2 en la cuál está la subrutina que atenderá la interrupción. Existe toda una zona de memoria P ej. A8 dónde se almacenan las direcciones1. Ya que el vector de dirección es un byte, A8 estará en otro registro delimitando la tabla, éste registro no existe en Indalo III. sino que se guardan en memoria a partir de la posición 00 00.

La tabla ocupará 512 bytes para direccionar 256 direcciones diferentes. En los pares está el primer byte de la posición y en las impares el segundo. El vector, entonces, será multiplicado por 2, para lo cuál está el MPX debajo del registro DIRL, DIRH será

0000 0000 y nos dará la dirección1.

Interrupción Hardware en Indalo III

Un interface ha activado la orden INT, se atiende si las interrupciones no están enmascaradas (FI=1), sino no se atienden.

S no están enmascaradas se termina la instrucción en curso y se activa INTA, y en el siguiente ciclo de reloj se lee el bus de datos dónde está el vector que se carga en DIRL.

Interrupción Software

INT vector Es un byte que va a continuación de su CO

Estas interrupciones se ejecutan siempre, independientemente del FI.

Común a software y hardware

Se almacena el PC en la pila como si fuese una subrutina. Se calcula dirección1, se genera un ciclo de lectura de dirección1. El dato viene por el Dbus que es dirección2 y que se carga en PC. La interrupción también acaba en RET.

INSTRUCCIONES DE INTERRUPCIONES DE INDALO III

CLI Pone el Flag de interrupción a 0

STI Pone el Flag de interrupción a 1.

INT vector Interrupción software.

HLT Detiene la CPU al ejecutarla.

Cuándo se detiene, sólo se pone en marcha con la llegada de una interrupción hardware.

Es conveniente hacer CLI dentro del código de atención a subrutina y guardar F para que al volver deje el Flag de interrupciones como estaba.

6.6 ACCESO DIRECTO A MEMORIA (DMA)

Capacidad de permitir la transferencia de información entre interface y memoria sin que intervenga el propio micro.

El periférico dice a la CPU que quiere tener DMA mediante la línea HOLD. Necesitará tener acceso al Abus y al Dbus. La CPU devuelve la microorden **holda** en reconocimiento a esa petición facilitándole el control de los buses mediante las puertas triestado del bus de control. Durante el DMA, el interface del periférico coloca en Abus una dirección de memoria y en el bus de control las señales de control de lectura ó escritura rd/wr (ciclo de máquina dónde no interviene la CPU). La CPU se mantiene inoperante.

6.7 PUERTOS DE E/S

La CPU se comunica con el exterior mediante unos registros llamados puertos de E/S. Con estos registros se pretende el poder leer y escribir en los periféricos como si fuese un acceso a memoria. Al generar ciclos de lectura ó escritura en puertos, estamos generando otro tipo de ciclos de máquina.

Si los puertos están en memoria se llama **mapeada en memoria**, dónde se reserva un espacio para registrar estas posiciones, la CPU no sabe si está escribiendo en memoria ó en un puerto. Si están en direcciones separadas de memoria se llama **directamente mapeada**, que es la utilizada en Indalo III, ó se accede a uno ó a otro.

La línea IO indica que vamos a direccionar el bloque de puertos. El mapa de puertos de E/S tiene 256 posiciones, se direcciona por tanto con 8 bits, para lo que están las puertas AND del Abus que desactivan 8 líneas si MEM está desactivada. El direccionamiento es directo por registro (se accede a la dirección que apunta el registro). Sólo funcionan con el registro C. La órdenes son:

IN A,(C) Lee al acumulador el puerto de la dirección en C.

OUT (C), A Escribe en el puerto de la dirección en C, el contenido de A.

TEMA 7. INTERFACES DE ENTRADA / SALIDA

7.1 INTRODUCCION

Los periféricos pueden utilizar diferentes corrientes e ir a distintas velocidades que la CPU, por lo que están los interfaces capaces de enlazar la CPU con los periféricos. Las líneas que utiliza Indalo III son IO, WR y RD.

7.2 CERROJOS Y PUERTAS TRIESTADO

Los buses tienen que quedar aislados cuando se están utilizando los dispositivos externos. Las puertas triestado se activan por nivel y su entrada de control se pude llamar OE (Output Enable) ó G(Gate).

Los cerrojos (latch) están porque es necesario mantener los datos durante cierto tiempo para que al exterior le dé tiempo a recogerlos; Son activos por flanco.

Indalo III sólo puede tener 16 puertos porque el decodificador sólo decodifica 4 líneas (las menos significativas del Abus). Se dice que tiene una decodificación incompleta. Por ejemplo: xxxx 0110

7.3 ACTIVACION DE LOS CIRCUITOS DE ENTRADA/SALIDA

Para activar un circuito de E/S lo primero que hay que hacer es direccionarlo (poniendo una dirección en Abus) y sincronizar con el bus de control el momento exacto en que se quiere leer ó escribir.

Decodificación centralizada: Utiliza decodificadores a los que le entran a0 a1 a2 a3 y seleccionan el puerto a activar. Se utiliza cuándo hay muchos puertos a direccionar.

Decodificación distribuida: Puerto a puerto, dirección a dirección se van seleccionando mediante puertas AND. Se utiliza para direccionar pocos puertos.

7.4 CONTROL DE LA IMPRESORA. INTERFACE CENTRONICS

Es un puerto paralelo utilizado para el control de la impresora. Tiene 8 líneas de datos en paralelo. *Tabla de la página 5 de las fotocopias.*

La línea **BUSY** se activa por cuatro razones:

*Por procesar un carácter que se acaba de recibir.

*Por que se está imprimiendo.

*La impresora está OFF−LINE.

*Se ha producido un error.

La línea **ERROR** se activa:

*Cuando está OFF−LINE.

*Cuando está sin papel.

La gestión de la impresora puede ser por interrupciones ó por subrutinas.

GESTION DE LA IMPRESORA POR SUBRUTINAS

Mirando la línea BUSY, y si está activa se sigue mirando en un bucle hasta que se desactive. Si no está activa, podemos empezar a enviar datos. Primero se envía un carácter y se activa STROBE (0.5 s), se esperan otros 0.5 s y se desactiva STROBE esperando otros 0.5 s , para regresar al programa llamador.

La subrutina INIC_IMP inicializa la impresora (página 6 fotocopias).

TEST_IMP comprueba el estado de la impresora (página 7).

GESTION DE LA IMPRESORA POR INTERRUPCIONES

Se une la línea ACK del puerto con la línea INT de la CPU.

Existen unos controladores de interrupciones programables llamadas PIC.

Pueden programarse para habilitar ó deshabilitar individualmente cada una de las interrupciones. Pueden asignarse prioridades, de forma que se atienda la interrupción en un orden establecido. En la rutina IMPRIMIR se guarda en BC el número de caracteres a imprimir y en X las direcciones de memoria de los caracteres. BC se guarda en la posición 200H y 201H ya que hasta la 1FF está la tabla de direcciones de atención a las interrupciones, y X lo guarda en la 202H y 203H. El programa hace una llamada a impresión y se imprimen un número de caracteres situados en memoria.

CALL HAB_3 Habilita la interrupción 3.

CALL DESHAB_3 La deshabilita.

Se puede decir que CALL INT_IMPR es lo mismo que INT 3 ya que ésta última indica la posición en la que está la subrutina INT_IMPR.

XXXI