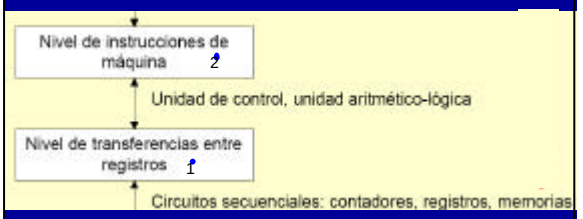
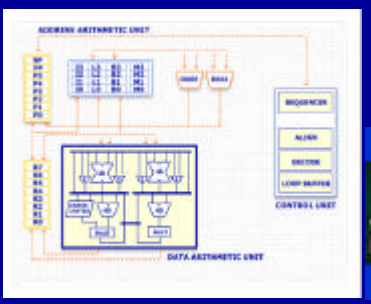
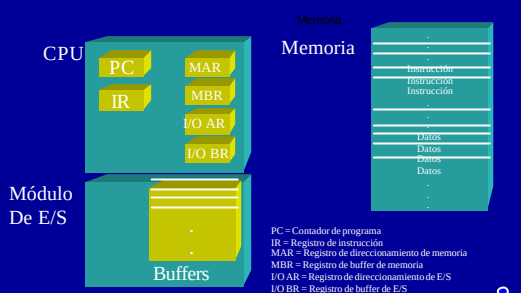
Nivel de Micromáquina (Microprogramación)



Circuitos Secuenciales : Contadores, Registros, memorias.

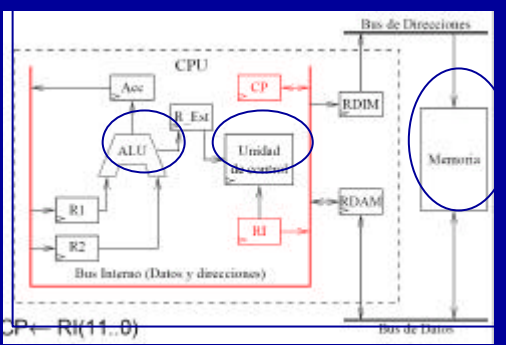


Componentes de la Computadora: Nivel de transferencia entre registros



PC = Contador de programa
IR = Registro de instrucción
MAR = Registro de direccionamiento de memoria
MBR = Registro de buffer de memoria
I/O AR = Registro de direccionamiento de E/S
I/O BR = Registro de buffer de E/S

Unidad de Control, Unidad Aritmética Lógica



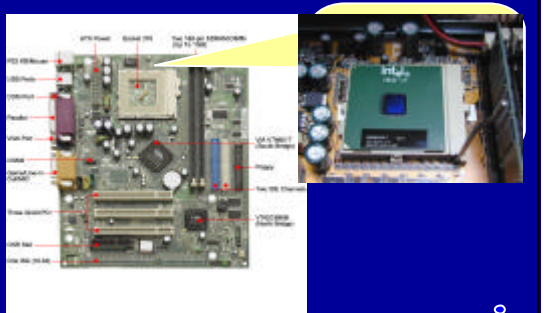
Función de la unidad de control

- Por cada operación se provee un código único
  - Ejemplos:
    - ADD
    - MOVE
- Un segmento de hardware que acepta el código y emite las señales de control

Componentes

- La UDC y la ALU constituyen el CPU.
- Necesitan entrar al sistema datos e instrucciones y los resultados salir:
  - Entrada/Salida
- Se necesita almacenamiento temporal de código y resultados:
  - Memoria principal

Nivel de Micromáquina (Microprogramación)



Objetivo del Capítulo

I. Tener una visión y un contexto de la microprogramación.

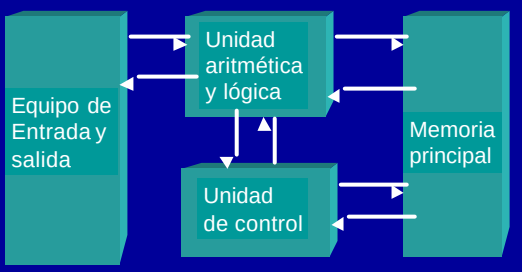
II. Desarrollar microprogamas.

III. Conocer la composición de las instrucciones y su secuencia.

Contenido

I. Capítulo
- Revisión de conceptos básicos.
- Estructura del CPU.
- Tipos de registros.
- El Ciclo de ejecución de las instrucciones.
- El ciclo de las interrupciones.
- Microprogramación
- Aplicaciones de la microprogramación.

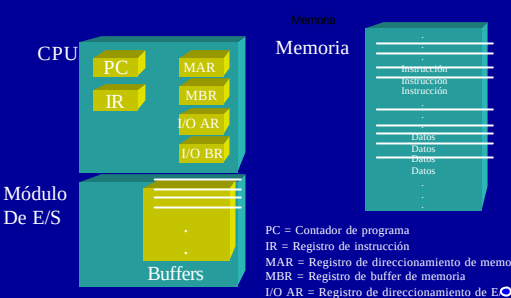
Estructura de la máquina de Von Neumann



von Neumann/Turing

- Concepto de programa almacenado
- Memoria principal almacenando programas y datos
- ALU operando datos binarios
- Unidad de control interpretando instrucciones de la memoria y ejecutándolas
- Equipo de entrada y salida operado por la unidad de control

Componentes de la Computadora: Nivel de transferencia entre registros



PC = Contador de programa
IR = Registro de instrucción
MAR = Registro de direccionamiento de memoria
MBR = Registro de buffer de memoria
I/O AR = Registro de direccionamiento de E/S
I/O BR = Registro de buffer de E/S

Estructura del CPU

- El CPU debe:
  - Extraer instrucciones
  - Interpretar instrucciones
  - Extraer datos
  - Procesar datos
  - Escribir datos

Registros

- El CPU debe tener un espacio de trabajo (almacenamiento temporal)
- Llamados registros
- La cantidad y función varía dependiendo del diseño del procesador.
- Una de las decisiones más importantes de diseño
- El nivel más alto en la jerarquía de memoria

Registros visibles al usuario

- Propósito general
- Datos
- Direcciones
- Códigos de condición

Registros de propósito general (1)

- Pueden ser verdaderamente de propósito general
- Pueden ser restringidos
- Pueden usarse para datos o direccionamiento
  - Datos
    - Acumulador
  - Direccionamiento
    - Segmento

Registros de propósito general (2)

- Hacerlos de propósito general
  - Aumentan la flexibilidad y opciones del programador
  - Aumenta el tamaño de la instrucción y complejidad
- Hacerlos especializados
  - Instrucciones más pequeñas (más rápidas)
  - Menor flexibilidad

¿Cuántos registros de propósito general?

- Entre 8 – 32
- Menos
  - Más referencias a memoria
- Más
  - No reducen notablemente las referencias a memoria.
- Ver RISC

¿Qué tan grandes?

- Lo suficientemente grandes para guardar una dirección completa.
- Lo suficientemente grandes para guardar una palabra completa.
- Tambien es posible combinar dos registros de datos
  - Programando en C

```
double int a;
long int a;
```


Registros de código de condición

- Conjunto de bits individuales
  - ejemplo. Si resultado de la última operación es cero
- Pueden leerse (implícitamente) por programas
  - ejemplo. Salta si es cero
- No se puede (normalmente) establecer por programas

Registros de estado y de control

- Contador del programa
- Registro decodificador de instrucción
- MAR = Registro de direccionamiento de memoria
- MBR = Registro de buffer de memoria
- ¿Qué hacen todos esos?

PSW (Palabra de estado del programa)

- Toda CPU incluye un registro o un conjunto de registros.
- Contiene un conjunto de bits
- Incluye códigos de condición
- Signo : del último resultado de la operación aritmética
- Cero : cuando el resultado da 0
- Acarreo (Carry) un digito (bit) de orden mayor (carry (addition) into or prestado (subtraction)
- Igualdad :Puesto cuando el resultado de una comparación lógica es igual .
- Sobreflujo indicar un (Overflow) aritmético
- Interrupciones : habilitadas o deshabilitadas
- Supervisor – Indica el modo en que corre: modo usuario o Supervisor

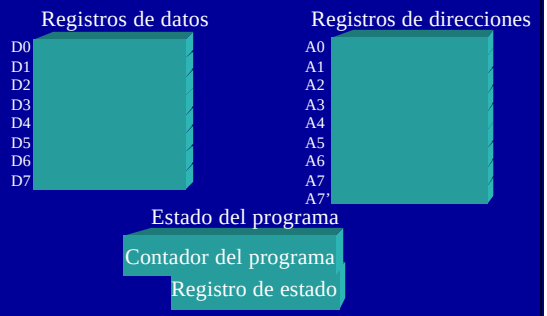
Modo Supervisor

- Intel ring zero
- Modo Kernel
- Permite que se ejecuten instrucciones Privilegiadas
- Usadas por el Sistema Operativo
- No disponibles para los programas de usuario

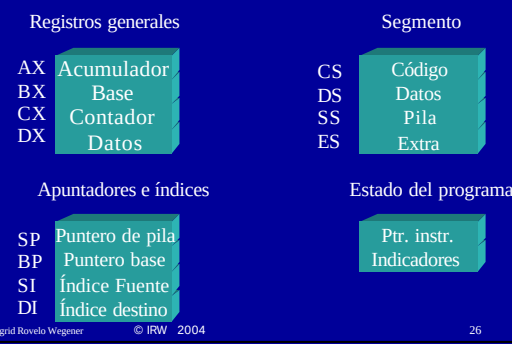
Otros registros

- Puede haber registros apuntando a:
  - PCBs (Bloques de control de procesos)
    - Ver S.O.
  - Vectores de interrupción
    - Ver S.O.
- El diseño del CPU y del Sistema Operativo están fuertemente encadenados

Ejemplos de organizaciones de registros (MC68000)



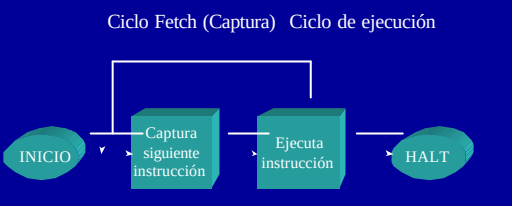
Ejemplos de organizaciones de registros (8086)



Ejemplos de organizaciones de registros (80386-Pentium)

Ejecución de una Instrucción

- Dos pasos:
  - Captura
  - Ejecución



Ciclo de captación

- El contador del programa (PC) tiene la dirección de la siguiente instrucción a captar
- El procesador capta la instrucción de la posición de memoria apuntada por PC
- Incrementa PC
  - Al menos que se le indique otra cosa
- La instrucción se carga en el registro de instrucción (IR)
- El Procesador interpreta la instrucción y ejecuta las acciones requeridas

Ciclo de ejecución

- Procesador-memoria
  - Transferencia de datos entre el CPU y la memoria principal
- Procesador E/S
  - Transferencia de datos entre CPU y un módulo de E/S
- Procesamiento de datos
  - Alguna operación aritmética o lógica en los datos
- Control
  - Alteración de la secuencia de las operaciones
  - ejemplo. jump
- Combinación de los de arriba

Ejemplo de la ejecución de un programa

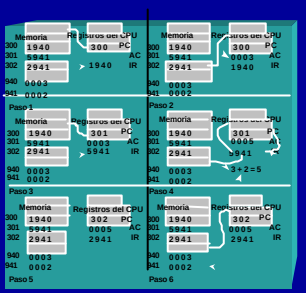
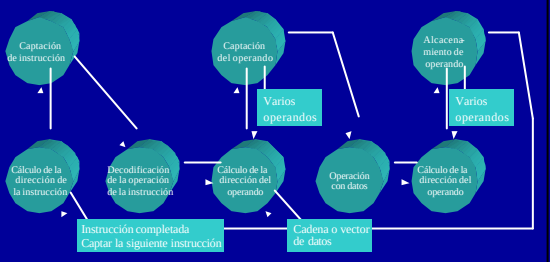


Diagrama de estados del ciclo de instrucción



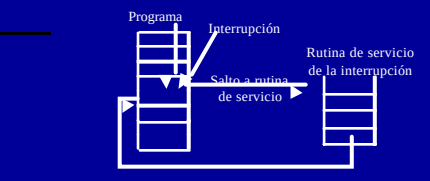
Excepciones e interrupciones

- **Definiciones**
  - Eventos inesperados que cambian el flujo normal de ejecución de las instrucciones
  - Excepción
    - Evento que tiene su origen en el interior del procesador (desbordamiento aritmético, instrucción ilegal, etc.)
  - Interrupción
    - Evento que tiene su origen en el exterior del procesador (dispositivos de entrada/salida, fallo de página, etc.)

Tipo de evento	¿Desde dónde?	Terminología MIPS
Petición dipositivo E/S	Exterior	Interrupción
Invocar Sistema Operativo desde el programa de usuario	Interior	Excepción
Desbordamiento aritmético	Interior	Excepción
Instrucción ilegal o no definida	Interior	Excepción
Fallo hardware	Cualquiera	Excepción/interrupción

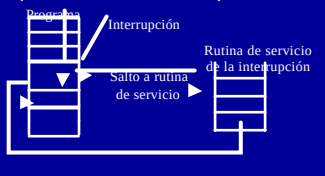
Tratamiento de excepciones en MIPS

- Las acciones básicas a realizar son:
  - Guardar la dirección de la instrucción causante en el registro “Contador de Programa de Excepciones (EPC)
  - Transferir el control al Sistema Operativo en alguna dirección especificada donde se tratará la excepción (ejecución de una rutina de servicio)



Tratamiento de excepciones en MIPS

- Información sobre la causa de la excepción (identificación de la excepción)
  - Registro CAUSE (en MIPS)
  - En caso de interrupciones vectorizadas (en otros procesadores)
    - La dirección a la que se transfiere el control está determinada por la causa de la excepción.



Implementación de excepciones en MIPS

- Excepciones a implementar
  - Desbordamiento aritmético
  - Instrucción ilegal o no definida
- Registros adicionales requeridos
  - **EPC**: Registro de 32 bits para guardar la dirección de la instrucción causante de la excepción
  - **CAUSE:** Registro de 32 bits para registrar la causa de la excepción. Utilizaremos sólo el bit menos significativo
    - bit 0 = 0 -> Instrucción ilegal
    - bit 0 = 1 -> Desbordamiento aritmético

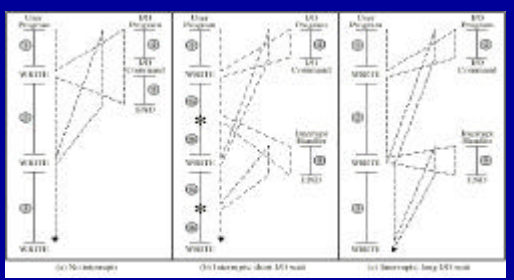
Implementación de excepciones en MIPS

- Señales de control adicionales
  - **Intcause** (0: instrucción ilegal - 1: desbordamiento)
  - **CauseWrite** (1: escritura en el registro CAUSE - 0: no escribe)
  - **EPCWrite** (1: escritura en el registro EPC - 0: no escribe)
  - **Constante**: C000 0000 0000 0000 (dirección a donde se transfiere el control cada vez que se interrumpe)

Interrupciones

- Mecanismo por el cual otros módulos (ej E/S) pueden interrumpir la secuencia normal de procesamiento
- Programa
  - ejemplo. overflow, división entre cero
- Timer
  - Generado por el reloj interno del procesador
  - Usado en la expropiación multi-tarea
- E/S
  - De un controlador de E/S
- Falla del Hardware
  - Error de paridad en la memoria

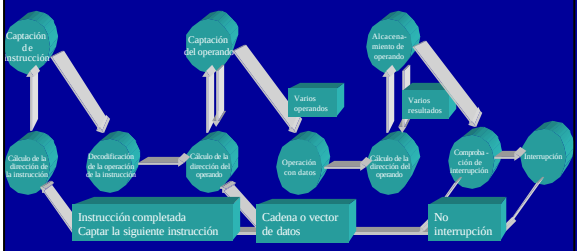
Control de flujo del programa



Ciclo de interrupción

- Se añade al ciclo de instrucción
- El procesador checa si hay interrupciones
  - Indicada por una señal de interrupción
- Si no hay interrupción, extrae la siguiente instrucción
- Si hay una interrupción pendiente:
  - Suspende la ejecución del programa actual
  - Guarda el contexto
  - Establece PC a la dirección de inicio de la rutina de servicio de interrupción
  - Interrupción del proceso
  - Restaura el contexto y continua con el programa interrumpido

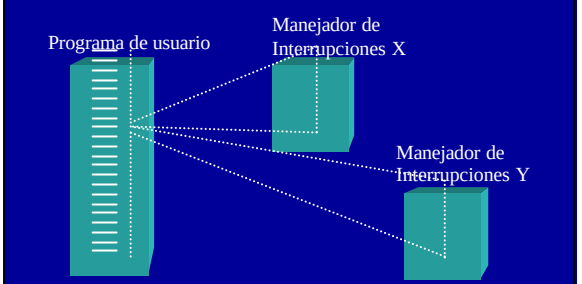
Diagrama de estados del ciclo de instrucción (con interrupción)



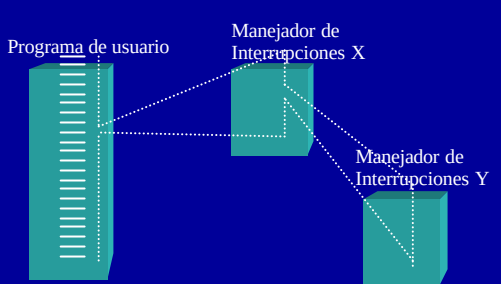
Interrupciones Múltiples

- Deshabilitar las interrupciones
  - El procesador ignorará otras interrupciones mientras procesa una interrupción
  - Las interrupciones permanecen pendientes y se checan una vez que se procesó la primera interrupción
  - Las interrupciones se manejan en secuencia como van ocurriendo
- Define prioridades
  - Las interrupciones de baja prioridad pueden interrumpirse por interrupciones de mayor prioridad
  - Cuando se termina de procesar una interrupción de más alta prioridad se regresa a la interrupción previa

Procesamiento de una secuencia de interrupciones



Procesamiento de interrupciones anidadas



Ciclo indirecto

- Puede requerir acceso a memoria para extraer operandos
- El direccionamiento indirecto requiere más accesos a memoria
- Puede ser a través de, o como un subciclo de instrucción adicional

El ciclo de instrucción

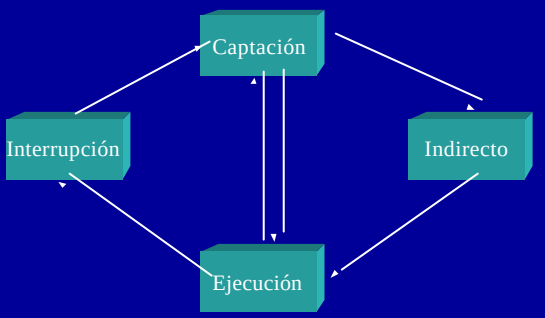


Diagrama de estados del ciclo de instrucción

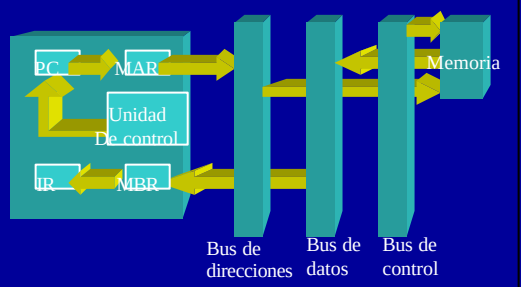
Flujo de datos (Extracción de la instrucción)

- Depende del diseño del CPU
- En general:
- Extraer
  - PC contiene la dirección de la siguiente instrucción
  - La dirección se mueve a MAR
  - Se pone la dirección en el bus de direcciones
  - La unidad de control solicita lectura a memoria
  - El resultado se pone en el bus de datos, copiado a MBR, y después a IR
  - Mientras tanto PC se incrementa a 1

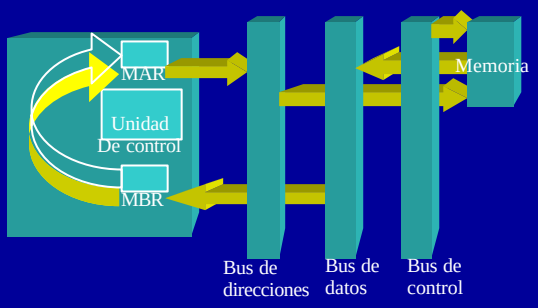
Flujo de datos (Extracción de los datos)

- Se examina IR
- Si es direccionamiento indirecto, se ejecuta el ciclo indirecto:
  - Los N bits más de la derecha de MBR se transfieren a MAR.
  - La unidad de control solicita lectura de memoria.
  - El resultado (dirección del operando) se mueve a MBR.

Flujo de datos, ciclo de captación



Flujo de datos, ciclo indirecto



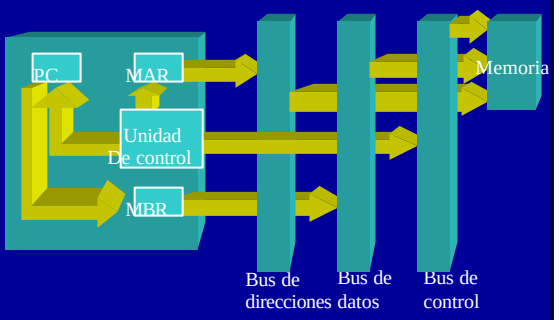
Flujo de datos (Ejecución)

- Puede tomar varias formas
- Depende de la instrucción que se está ejecutando
- Puede incluir :
  - Lectura/Escritura de la memoria
  - Entrada/Salida
  - Transferencias de registros
  - Operaciones en la ALU

Flujo de datos (Interrupción)

- Simple
- Predecible
- El PC actual se guarda para poder continuar después de la interrupción.
- El contenido de PC se copia a MBR
- Una dirección de memoria especial se carga en MAR
  - Ejemplo. Apuntador de la pila (Stack Pointer)
- MBR se escribe en memoria
- Se carga PC con la dirección de manejo de la interrupción
- La siguiente instrucción (primera del manejador de interrupciones) puede extraerse.

Flujo de datos, ciclo de interrupción



Pre-captura

- La captura se hace accesando memoria.
- La ejecución usualmente no accesa memoria principal.
- Puede capturar la siguiente instrucción durante la ejecución de la instrucción actual.
- A esto se le llama pre-captura de instrucción.

Mejora el rendimiento

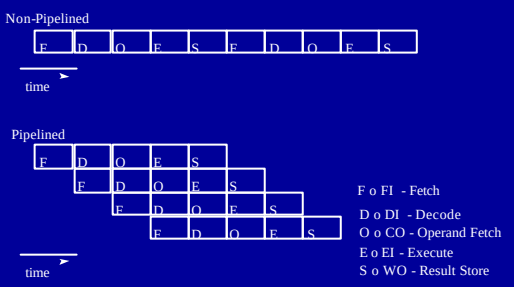
- Pero no al doble:
  - La captura es normalmente más corta que la ejecución
    - ¿Pre-capturar más de una instrucción?
  - Cualquier salto significa que las instrucciones pre-capturadas no se requieren
- Añade más estapas para mejorar el rendimiento

Procesamiento en tuberías (Pipelining)

- Extraer instrucción
- Decodifica instrucción
- Calcula operandos
- Extraer operandos
- Ejecutar instrucciones
- Escribir resultados

- Sobreponer estas operaciones

Procesamiento en tuberías (Pipelining)



Recap: Instruction Cycle

- **F**etch - fetch the instruction from the memory
- **D**ecode - decode the instruction
- **O**perand Fetch - get the necessary operands
- **E**xecute - execute the instruction
- **S**tore - store the result to the appropriate location

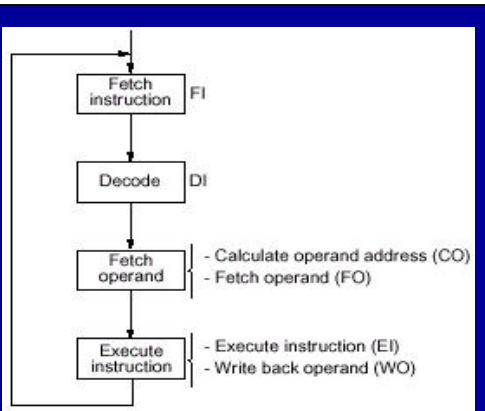
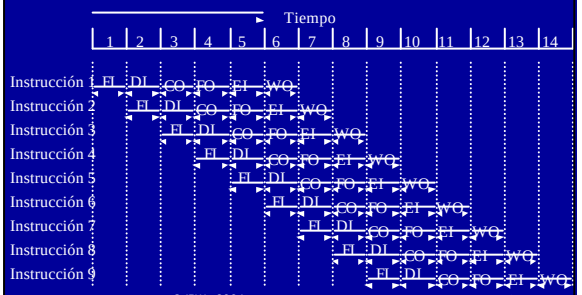
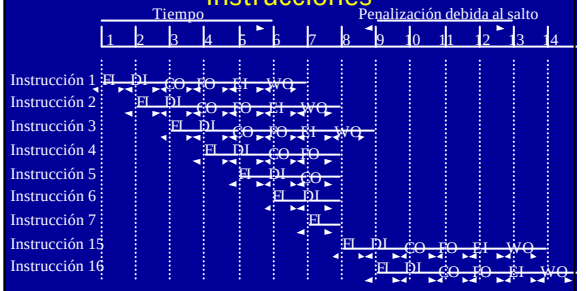


Diagrama de tiempos del funcionamiento del cauce de instrucciones



Efecto de un salto condicional en el funcionamiento del cauce de instrucciones



Porque tuberías (Pipelining) ?

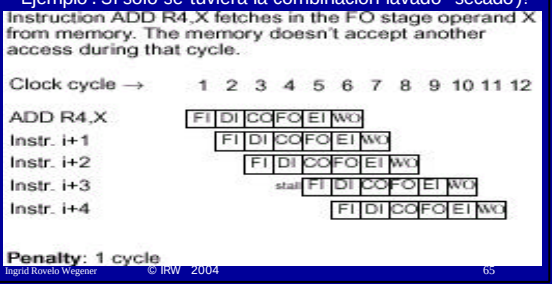
- La idea básica es dividir el camino de datos dentro de etapas.
- De los recursos que esten listos y disponibles en la máquina.

Riesgos y obstáculos

- Los obstáculos (Hazards) son situaciones que previenen a la siguiente instrucción sea ejecutada (demorada) durante el turno designado y tiende a reducir el rendimiento ganado de las tuberías.
- Cuando una instrucción es demorada, todas las instrucciones posteriores a la instrucción demorada en la tubería también son demoradas
- La Instrucción anterior a la demorada puede continuar
- Ninguna instrucción nueva es captada durante la demora
- 3 clases de obstáculos:
  - estructurales
  - datos
  - control

Obstáculos estructurales

- **Intento de utilización del mismo recursos (memoria / unidad funcional) al mismo tiempo por más de una instrucción**
  Ejemplo . Si solo se tuviera la combinación lavado- secado)?



Obstáculos estructurales

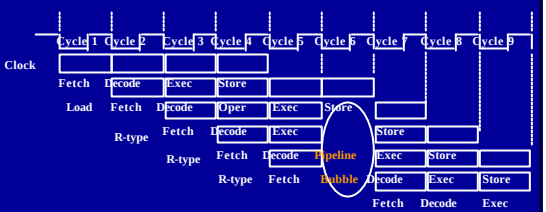
- Causas: Recursos (usualmente hardware) conflictos entre el translape de instrucciones.
- 2 tipos:
  - Causado por el conflicto de un recurso no hecho redundante para la tubería .
  - Causado por el conflicto de un recurso hecho redundante para la tubería pero no habilitado a enfrentarse con el translape.

Demora

- La solución general a los obstáculos en las tuberías es demorar la tubería, esto es prevenir que la instrucción sucesiva en la tubería realice su fase en el ciclo.
- Esto permite que la instrucción en cuestión proceda sin conflicto pero arruinando el translape de las intrucciones para una fase.

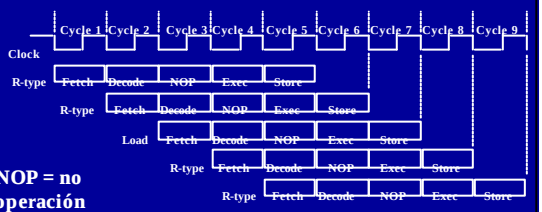
Solución 1: Una burbuja en la Tubería

- Insertar una "bubuja" dentro de la tubería para prevenir 2 *escrituras* o *almacenamientos* el mismo ciyclo.



Solución 2: Alargar (Dilatar) el ciclo de Almacenamiento

- Dilatar la operación por medio de la adición de un estado de *NOP*



NOP = no operación

Obstáculos de Datos

- Los conflictos en los datos del programa que se dan lugar por las dependencias entre las instrucciones.
- 3 tipos:
  - RAW (Read After Write)
  - WAR (Write After Read)
  - WAW (Write After Write)

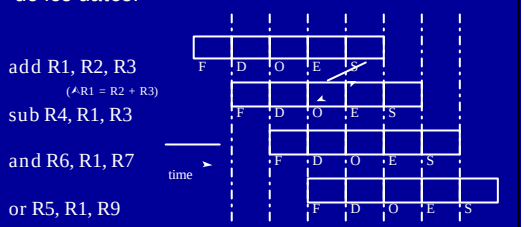
Obtáculos de Datos

- Intentan utilizar un elemento antes de que esté listo.

➔Las instrucciones dependen de un resultado anterior de una instrucción que sigue en la tubería.

Dependencia de los Datos

- Las dependencias de los Datos son obstáculos de los datos.

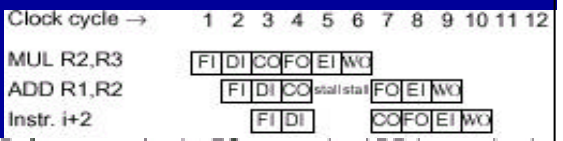


read-after-write (RAW) hazard

Solución: Seguro interno en la Tubería Pipeline Interlock

- Un seguro interno en la tubería es un hardware especial utilizado para detectar las dependencias de los datos en la tubería y que puedan causar la demora.

I1:	MUL	R2,R3	R2 ← R2 * R3
I2:	ADD	R1,R2	R1 ← R1 + R2

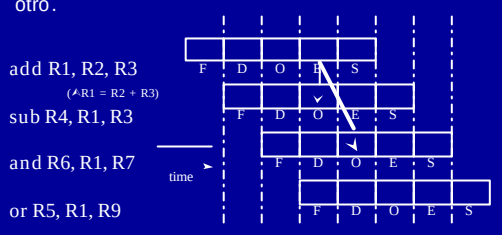


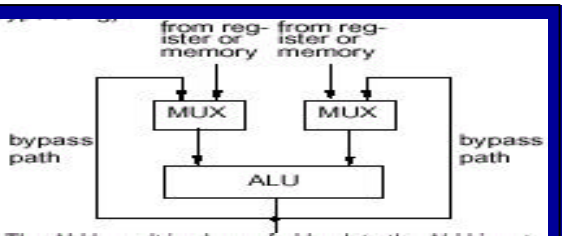
Before executing its FO stage, the ADD instruction is stalled until the MUL instruction has written the result into R2.

Penalty: 2 cycles

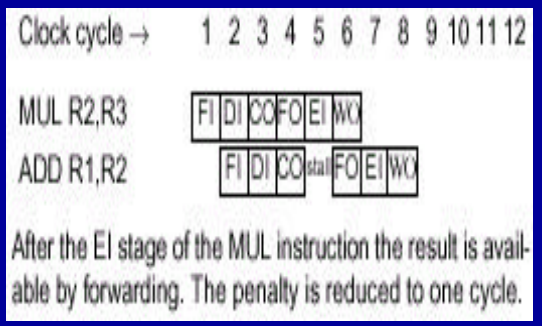
Solución: Adelantarse (Bypassing)

- "Adelantarse " (Forward) el resultado de un estado a otro.





The ALU result is always fed back to the ALU input. If the hardware detects that the value needed for the current operation is the one produced by the previous operation (but which has not yet been written back) it selects the forwarded result as the ALU input, instead of the value read from register or memory.



Adelantándose (Forwarding)

- La demora deberá ser utilizado para resolver los obstáculos de los datos pero deberá ser evitado cuando sea posible.
- Si la fuente de la instrucción en curso es la misma como el destino de una temprana (anterior) deberá utilizarse el adelanto.
- El camino de datos permite a los datos el ser adelantados a la siguiente instrucción (como el fuente) antes de ser escrito a su destino.

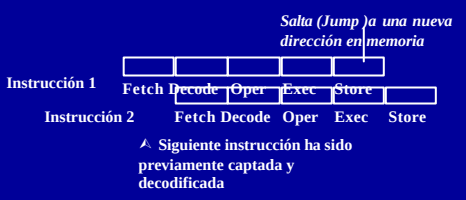
Calendarización de la Tubería

- Para prevenir los obstáculos en los datos, los compiladores deberán modificar el código compilado (en el lenguaje de máquina para evitar la demora.

Obstáculos de Control

- Intento de hacer una decisión antes de que una condición sea evaluada.

➔ Instrucción de salto (branch)



Obstáculos de Control

- La ramificación (o salto) del programa (jumps) puede causar una forma de demora en la tubería porque antes de que al PC se le escriba con la dirección destino una nueva instrucción ha sido previamente capturada.

- Solución:
  - Determinar si en la instrucción tendrá a lugar una ramificación (o salto) (el valor del PC es cambiado a la dirección meta) antes de que sea manejada.
  - Computa la dirección meta en avanzada.

Tratamiento de saltos

- Múltiples flujos
- Precaptar el destino del salto
- Buffer de ciclos – almacenamiento intermedio
- Predicción de saltos
- Salto retardado

Flujos múltiples

- Duplicar las partes iniciales del cauce
- Dejar que capte las dos instrucciones utilizando los dos caminos
- Hay retardos debidos a la competencia por el acceso a los registros y memoria
- Varios saltos nos llevan a que se necesitan más tuberías

Precaptar el destino del salto

- Se precapta la instrucción destino del salto además de la siguiente a la bifurcación
- Se guarda esta instrucción hasta que se ejecute la instrucción de bifurcación
- IBM 360/91 usa este método

Buffer de ciclos

- Memoria muy rápida
- Gestionada por la etapa de captación de instrucción
- Checa el buffer antes de captar de la memoria
- Muy bueno para saltos o ciclos pequeños
- Como un cache
- Usada por la CRAY-1

Predicción de saltos (1)

- Predecir que nunca se salta
  - Asume que el salto no sucederá
  - Siempre captura la siguiente instrucción
  - 68020 y VAX 11/780
  - La VAX no pre-capturará después de un sato si resulta un fallo de página (diseño del SO v CPU)
- Predecir que siempre se salta
  - Asume que el salto ocurrirá
  - Siempre captura la instrucción destino

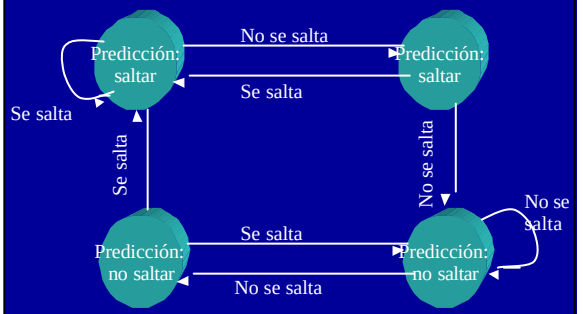
Predicción de saltos (2)

- Predecir según el código de operación
  - Algunas instrucciones parecen dar más el salto como resultado que otras
  - Tasas de acierto son como del 75 %
- Conmutador Saltar/No Saltar
  - Se basa en la historia previa
  - Buena para los ciclos

Predicción de saltos (3)

- Salto retardado
  - No se hace el salto hasta que no se tiene que hacer
  - Re-arregla instrucciones

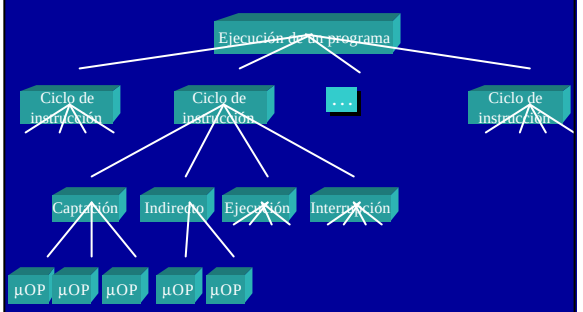
Diagrama de estados de la predicción de saltos



Micro-Operaciones

- Una computadora ejecuta un programa
- Ciclo de captación/ejecución
- Cada ciclo tiene un número de etapas
  - Ver pipelining
- Llamadas micro-operaciones
- Cada paso hace muy poco
- Operación atómica del CPU

Elementos que constituyen un programa en ejecución



Captación - 4 Registros

- Memory Address Register (MAR)
  - Conectado al bus de direcciones
  - Especifica la dirección para una operación de lectura/escritura
- Memory Buffer Register (MBR)
  - Conectado al bus de datos
  - Tiene los datos a escribir o los datos leidos en la última lectura
- Program Counter (PC)
  - Tiene la dirección de la siguiente instrucción a ser captada
- Instruction Register (IR)
  - Tiene la última instrucción captada

Secuencia de captación

- La dirección de la siguiente instrucción está en PC
- La dirección (MAR) se pone en el bus de direcciones
- La unidad de control emite un comando de lectura
- Los resultados (datos de la memoria) aparecen en el bus de datos
- Los datos del bus de datos se copian en MBR
- PC se incrementa en 1 (en paralelo con los datos captados de la memoria)
- Los datos (instrucciones) se mueven de MBR a IR
- MBR queda libre para próximas captaciones de datos

Secuencia de captación (simbólica)

- t1: MAR <- (PC)
- t2: MBR <- (memoria)
- PC <- (PC) +1
- t3: IR <- (MBR)
- (tx = unidad de tiempo/ciclo de reloj)
- o
- t1: MAR <- (PC)
- t2: MBR <- (memoria)
- t3: PC <- (PC) +1
- IR <- (MBR)

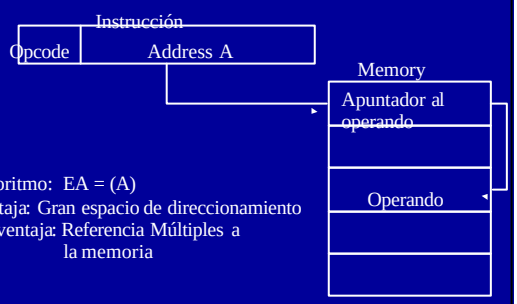
Reglas para agrupar ciclos de reloj

- Debe seguirse la secuencia correcta de eventos
  - MAR <- (PC) debe preceder MBR <- (memoria)
- Deben evitarse conflictos
  - No se debe leer y escribir el mismo registro al mismo tiempo
  - MBR <- (memoria) e IR <- (MBR) no deben de estar en el mismo ciclo
- También: PC <- (PC) +1 es una suma
  - Usa la ALU
  - Puede necesitar micro-operaciones adicionales

Ciclo indirecto

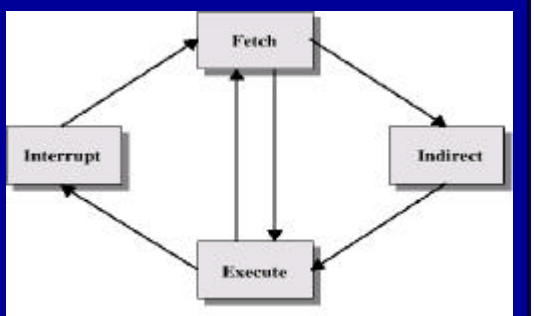
- MAR <- ($IR_{dirección}$) - campo dirección de IR
- MBR <- (memoria)
- $IR_{dirección}$ <- ($MBR_{dirección}$)
- MBR contiene una dirección
- IR está en el mismo estado que si no se hubiera usado direccionamiento indirecto
- (¿Qué dice esto sobre el tamaño de IR?)

Diagrama de direccionamiento Indirecto



Algoritmo: EA = (A)
Ventaja: Gran espacio de direccionamiento
Desventaja: Referencia Múltiples a la memoria

Ciclo de instrucción indirecto



Ciclo de interrupción

- t1: MBR <-(PC)
- t2: MAR <- dirección de salvaguarda
- PC <- dirección de la rutina
- t3: memoria <- (MBR)
- Esto es un mínimo
  - Puede haber micro-operaciones adicionales para obtener la dirección
  - Guardar el contexto lo hace la rutina de manejo de interrupción, no las micro-operaciones

Ciclo de ejecución (ADD)

- Diferente para cada instrucción
- ej. ADD R1,X – añade el contenido de la dirección X al registro 1, resultado en R1
- t1: MAR <- ($IR_{dirección}$)
- t2: MBR <- (memoria)
- t3: R1 <- R1 + (MBR)

Ciclo de ejecución (ISZ)

- ISZ X - incrementa y salta si es cero
  - t1: MAR <- ($IR_{dirección}$)
  - t2: MBR <- (memoria)
  - t3: MBR <- (MBR) + 1
  - t4: memoria <- (MBR)
  - if (MBR) == 0 then PC <- (PC) + 1
- Notes:
  - La comprobación y actuación puede implantarse como una microoperación
  - Esta puede ejecutarse durante la misma unidad de tiempo en la cual el valor actualizado de MBR se guarda en memoria

Ciclo de ejecución (BSA)

- BSA X – Salta y guarda la dirección
  - La dirección de la instrucción que viene a continuación de la instrucción BSA, se guarda en la posición X
  - La ejecución continua de X+1
  - t1: MAR <- ($IR_{dirección}$)
  - MBR <- (PC)
  - t2: PC <- ($IR_{dirección}$)
  - memoria <- (MBR)
  - t3: PC <- (PC) + 1

Requisitos funcionales

- Definir los elementos básicos del procesador.
- Describir las microoperaciones que ejecuta el procesador.
- Determinar las funciones que debe realizar la unidad de control.

Elementos básicos del procesador

- ALU
- Registros
- Caminos de datos internos
- Caminos de datos externos
- Unidad de control

Tipos de microoperación

- Transferencia de datos entre registros
- Transferencia de datos de registro a externo
- Transferencia de datos de externo a registro
- Hacer operaciones aritméticas o lógicas

Funciones de la unidad de control

- Secuenciamiento
  - Hace que el CPU avance a través de una serie de microoperaciones
- Ejecución
  - Hace que se ejecute cada microoperación
- Esto se hace usando señales de control

Señales de control (1)

- Reloj
  - Una micro-instrucción (o conjunto de micro-instrucciones paralelas) por ciclo de reloj
- Registro de instrucción
  - Op-code para la instrucción actual
  - Determina que micro-instrucciones se ejecutan

Señales de control (2)

- Banderas
  - Estado del CPU
  - Resultado de operaciones previas
- Del bus de control
  - Interrupciones
  - Reconocimiento

Señales de control - salidas

- Internas al CPU
  - Causan movimiento de datos
  - Activan funciones específicas de la ALU
- Via bus de control
  - A memoria
  - A módulos de E/S

Ejemplo de señales de control - Captación

- MAR <- (PC)
  - La unidad de control activa una señal que abre las puertas entre MAR y PC
- MBR <- (memoria)
  - Abre las puertas entre MAR y el bus de direcciones
  - La memoria lee señales de control
  - Abre las puertas entre el bus de datos y MBR

Organización interna

- Usualmente un solo bus interno
- Las puertas controlan el movimiento de datos dentro y fuera del bus
- Las señales de control controlan la transferencia de hacia y desde el bus externo del sistema
- Se añaden dos registros temporales para el funcionamiento correcto de la ALU

Implementación cableada (1)

- Entradas de la unidad de control
- Las banderas y el bus de control
  - Cada bit tiene un significado
- Registro de instrucción
  - El Op-code causa señales de control diferentes por cada instrucción
  - Por cada op-code hay una lógica única
  - El decodificador toma la entrada codificada y produce una sola salida
  - *n* entradas binarias y 2^n salidas

Implementación cableada (2)

- Reloj
  - Emite una secuencia repetitiva de pulsos
  - Útil para medir la duración de las microoperaciones
  - Debe ser lo suficientemente largo para permitir la propagación de señales
  - Diferentes señales de control en diferentes tiempos dentro del ciclo de instrucción
  - Necesitan un contador con diferentes señales de control para t1, t2 etc.

Problemas con la implementación alambrada

- El secuenciamiento y la lógica de microoperaciones es compleja
- Difícil de diseñar y probar
- Diseño inflexible
- Difícil añadir nuevas instrucciones

Concepto de programa

- Los sistemas cableados son inflexibles
- El hardware de propósito general puede hacer diferentes taréas, dadas señales de control correctas
- En lugar de re-cablear, enviar un nuevo conjunto de señales de control.

¿Qué es un programa?

- Una secuencia de pasos.
- Por cada paso, se hace una operación aritmética o lógica
- Por cada operación, se requiere un conjunto diferente de señales de control

Control microprogramado

- Usa secuencias de instrucciones para controlar operaciones complejas
- Llamado micro-programación o firmware

Implementación (1)

- Todo lo que hace la unidad de control es generar un conjunto de señales de control
- Cada señal de control está encendida o apagada
- Representar cada señal de control por un bit
- Tiene una palabra de control por cada microoperación
- Tiene una secuencia de palabras de control por cada instrucción máquina
- Añade una dirección para especificar la siguiente micro-instrucción, dependiendo de condiciones

Implementación (2)

- Microprocesadores actuales
  - Muchas instrucciones y asociadas al nivel de registro del hardware
  - Muchos puntos de control a ser manipulados
- Esto resulta en memoria de control que
  - Contiene un número grande de palabras
    - Correspondiente al número de instrucciones a ser ejecutadas
  - Tiene un tamaño amplio de palabra
    - Debido al gran número de puntos de control a ser manipulados

Longitud de palabra de microprograma

- Basado en 3 factores
  - Número máximo de microoperaciones simultáneas soportado
  - La forma como la información de control es representada o codificada
  - La forma en la cual se especifica la siguiente dirección de microinstrucción

Tipos de microinstrucción

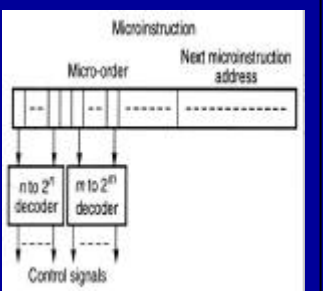
- Cada microinstrucción especifica una sola (o pocas) microoperaciones a ser ejecutadas
  - (microprogramación *vertical*)
- Cada microinstrucción especifica varias microoperaciones diferentes a ejecutarse en paralelo
  - (microprogramación *horizontal*)

Microprogramación

- Horizontal: cada microinstrucción contiene n bits, cada uno de los cuales controla directamente cada pieza de hardware, evitando otros niveles de decodificación. Las microinstrucciones horizontales son más largas.
- Vertical: cada microinstrucción tiene un pequeño número de campos, muy codificados, de forma que requieren más decodificación antes de aplicarse a los componentes individuales.

Microprogramación vertical

- Los Bits son codificados dentro de campos para reducir el número de bits necesarios en el micro-orden.
- Los Bits son formados en grupos especificando las señales que no se pueden activar al mismo tiempo.



Microprogramación vertical

- Tamaño pequeño del control de palabra.
  ➔ Más eficiente.
- Requiere lógica extra para decodificar patrones.
- Son necesarios más pasos para la ejecución.
  ➔ Reducida velocidad de operación.
- Menos flexible porque las señales mutuamente excluyentes tienen que se identificado y no se pueden introducir después nuevas combinaciones de señales.

Microprogramación vertical

- Son más compactas (ocupan menos bits)
- N señales de control condificadas en $\log_2 n$ bits
- Capacidad limitada para expresar paralelismo
- Información de codificación de control requiere una palabra decodificadora externa de memoria para identificar la línea de control exacta que está siendo manipulada.

Variaciones en la Microprogramación

° Microcódigo "Vertical"

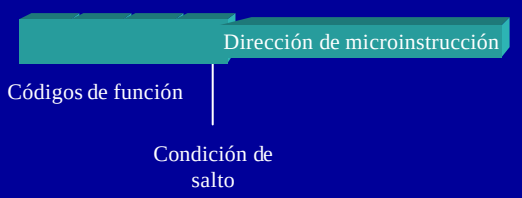
– compactar el formato de la microinstrucción para cada clase de microoperación

– decodificación local para generar todos los puntos de control.

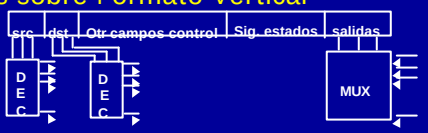
branch: µseq-op µadd
execute: ALU-op A,B,R
memory: mem-op S, D

Vertical

Microinstrucción vertical

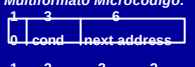


Más sobre Formato Vertical

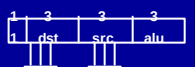


Algunos de éstos pueden no tener nada que hacer con los registros

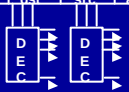
Multiformato Microcódigo:



Rama de salto -Branch Jump



Registro X tranfer. de Operación



Variaciones en la Microprogramación

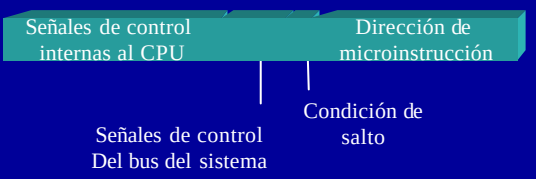
° Microcódigo "Horizontal"

– Un campo de control x cada punto de control en la máquina.

Microprogramación horizontal

- Palabra de direccionamiento de la memoria amplia
- Alto grado de operaciones paralelas posibles
- Poca codificación de información de control

Microinstrucción horizontal

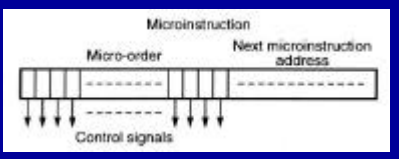


Microprogramación Horizontal

- Un bit es provisto para cada señal lógica que pueda ser genrada por la microinstrucción.
- Para generate una señal en particular, el bit correspondiente es puesto a 1.

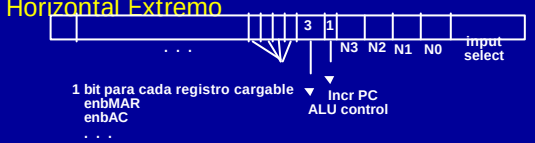
➔ Cada palabra de control es *n* bits de longitud donde *n* es el número total de señales de control en la máquina.

Microprogramación Horizontal



- Son necesarios menos pasos para la ejecución
- Más flexible.
- Gran tamaño de la palabra de control.

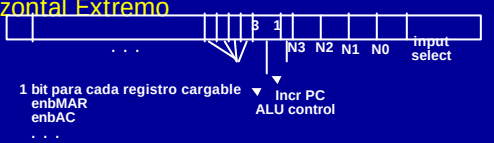
Horizontal Extremo



Dependiendo de la organización del bus, muchas de las combinaciones potenciales de control estarían mal (sin uso), porque implican transferencias que nunca pueden darse al mismo.

Tiene sentido codificarlas en campos a medida que ahorran espacio en ROM

Horizontal Extremo



Ejemplo: mem_to_reg y ALU_to_reg nunca deberán ocurrir simultáneamente;
=> codificar en en un solo bit el cual decodifica, en vez de hacerlo en dos bits separados

NOTE: Deberá mantenerse especificada en una sola microinstrucción la codificación suficiente en la medida que se puedan dar las acciones en paralelo que soporten el caminos de datos

Compromiso

- Divide las señales de control en grupos
- Implementa cada grupo como un campo separado en la palabra de memoria
- Soporta niveles razonables de paralelismo sin mucha complejidad

Microinstrucciones

Las máquinas con un control complejo pueden requerir gran cantidad de circuitería, Hay máquinas con :
- cientos o miles de microinstrucciones.
- microinstrucciones de cien o más bits.

Reducción del número de bits en la microinstrucción (codificación vertical):
- Codificación de campos:
  - Si un campo tiene **k** señales de control y en un ciclo de reloj dado sólo puede activarse una de ellas, pueden codificarse con **m** bits siendo **m** el menor entero que cumple que $2^m = k$.
  - Cada microinstrucción tendrá **m-k** bits menos.
  - A cambio, es preciso añadir un decodificador de **m** a 2^m.

Microinstrucciones

- Dos señales de control en el mismo campo codificado no pueden activarse simultáneamente: si fuese preciso hacerlo, se usarán dos microinstrucciones consecutivas, activándose cada señal en una de ellas.
- Definición de varios formatos de microinstrucción:
  - Los campos pueden solaparse unos con otros.
  - El significado de cada bit de la microinstrucción vendrá dado por un bit adicional que servirá para distinguir unos formatos de otros.
  - Será preciso añadir el bit de formato y un demultiplexor.
  - El número de microinstrucciones puede aumentar, si bien serán más estrechas.
- En ambos casos puede ser preciso aumentar el ciclo de reloj.

Codificación horizontal: codificación ancha con todos los bits de la microinstrucción.

Control Hibrido

No toda la información critica de control es derivada de la lógica de control

Ejem., Instruction Register (IR) contiene información útil de control, como los registros fuente, destino, y códigos de operación, etc.

Microprogramación Horizontal vs. Vertical

NOTA: Organizaciones previas no son microprogramación horizontal TRUE
El decodificador de instrucciones da los sabores de las microoperaciones codificadas

La mayoría de la microprogramación basada en el control varia entre:

La organización *horizontal* (1 bit de control por punto de control)

La organización *vertical* (campos codificados en la memoria de control y tienen que ser decodificados para controlar algo.)

Microprogramación Horizontal vs. Vertical

Horizontal

+ Mayor control sobre el paralelismo potencial de las operaciones en el camino de los datos.

- paralelismo de los usos hasta llenar el almacenamiento del control.

Vertical

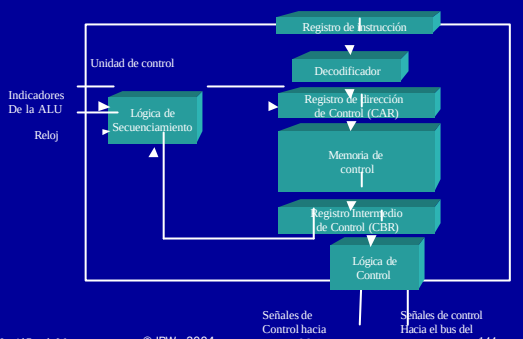
+ Fácil de programar, no es muy diferente a programar en una computradora RISC en lenguaje ensamblador.

- Nivel extra de decodificación puede bajar el rendimiento de la máquina.

Control de la Memoria

. Salta a indirecto o ejecución	Rutina de ciclo de captación
. Salta a ejecución	Rutina de ciclo indirecto
. Salta a captación	Rutina de ciclo de interrupción
Salta a rutina de Op code	Inicio del ciclo de ejecución
. Salta a captación o interrupción	Rutina AND
Salta a captación o interrupción	Rutina ADD

Funcionamiento de una unidad de control microprogramada



Función de la unidad de control

- La unidad de secuenciamiento carga el registro de dirección de control (CAR) y emite un comando de lectura
- El registro de dirección de control contiene la dirección de la siguiente microinstrucción a leer
- Cuando se lee una microinstrucción de la memoria de control, se transfiere al registro intermedio de control (CBR)

Función de la unidad de control

- El contenido del registro intermedio de control se conecta a las señales de control
  - Leer una microinstrucción es lo mismo que ejecutarla
- La lógica de secuenciamiento carga la nueva dirección en el registro de dirección de control basado en la información de la siguiente dirección del registro intermedio de control y las banderas de la ALU.

Ventajas y desventajas

- Simplifica el diseño de la unidad de control
  - Más barata
  - Menos propensa a errores
- Más lenta que el hardware

Tareas hechas por una unidad de control microprogramada

- Secuenciamiento de microinstrucciones
- Ejecución de microinstrucciones
- Deben considerarse ambas a la vez

Consideraciones de diseño

- Tamaño de las microinstrucciones
- Tiempo de generación de direcciones
  - Determinado por el registro de instrucción
    - Una por ciclo, después de que se captó la instrucción
  - Siguiente dirección secuencial
    - Lo más común
  - Saltos
    - Condicionales e incondicionales

Técnicas de secuenciamiento

- Basadas en
  - La microinstrucción actual
  - Banderas de condición
  - Contenidos del IR
  - La dirección de la memoria de control debe generarse
- Basado en el formato de la información de la dirección
  - Dos campos de dirección
  - Un campo de dirección
  - Formato variable

Generación de direcciones

Explícitas	Implícitas
Dos campos Salto incondicional Salto condicional	Traducción Adición Control Residual

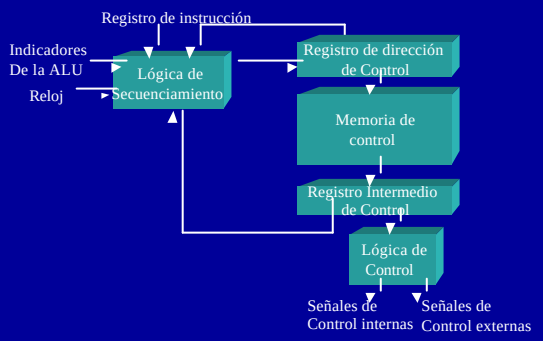
Ejecución

- El ciclo es el evento básico
- Cada ciclo está formado de dos eventos
  - Captación
    - Determinado por la generación de la dirección de la microinstrucción
  - Ejecución

Ejecución

- El efecto es generar señales de control
- Algunos puntos de control internos al procesador
- El resto va al bus de control u otra interfase

Organización de la unidad de control



Microprogramación: definiciones

•Microinstrucción: conjunto de valores de las señales de control en una etapa cualquiera de la ejecución de una instrucción.

• Ejecución de una microinstrucción: activación de las señales de control especificadas en la misma en un instante determinado.

• La ejecución de una instrucción implica la asignación de una serie de valores a las señales de control de forma ordenada en una secuencia de fases o etapas.

Microprogramación: definiciones

• La ejecución de una instrucción implica la ejecución ordenada de una secuencia de microinstrucciones.

• Es preciso establecer un mecanismo de secuenciamiento que permita decidir cuál es la microinstrucción siguiente a una dada.

• Las secuencias de microinstrucciones suelen estar ubicadas en forma contigua, unas detrás de otras.

Microprogramación: definiciones

• En ocasiones se producen "saltos" en la secuencia, y la microinstrucción que hay que ejecutar a continuación no es la que está detrás, sino otra distinta.

• Microprograma o microcódigo: conjunto de microinstrucciones utilizadas para especificar unidades de control de una máquina (para caminos de datos (mono y/o multiciclo y el control de los datos).

Microprogramación: definiciones

- Microprogramación: técnica de diseño del control como un microprograma encargado de realizar las instrucciones de la máquina en términos de microinstrucciones más simples.
- Es una estrategia particular para la implementación de la unidad de control de un procesador por medio de la programación de las operaciones a nivel de transferencia de registros

Microprogramación: definiciones

Idea clave: representar los valores activados en las líneas de control en forma simbólica.

• Así el microprograma es la representación de las microinstrucciones del mismo modo que el lenguaje ensamblador es una representación de las instrucciones de máquina.

Microprogramación: definiciones

• La sintaxis de la microprogramación tiene una cierta analogía a la sintaxis de la programación en ensamblador:

•La instrucciones de ensamblador se dividen en campos (etiqueta, código de operación, operandos, comentarios).

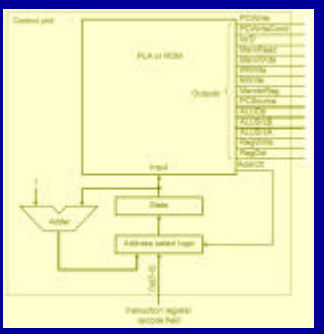
•La sintaxis de las microinstrucciones consta de varios campos o agrupaciones de señales.

•Los campos se definen de acuerdo a la función de las señales en el camino de datos.

Tareas Básicas Realizadas por una Unidad de Control Microprogramada

- El secuenciamiento de microinstrucciones
- La ejecución de microinstrucciones

Circuitos Secuenciales : Contadores, Registros, memorias.



Definición del formato de la Microinstrucción

- En una microinstrucción se debe poder especificar la activación de todas las señales generadas por la unidad de control.
- Las señales de control se pueden dividir en grupos (campos) dependiendo de la función que tengan dentro del camino de datos.
- El formato de microinstrucción debe simplificar la representación y hacer sencillo diseñar y comprender el microprograma.

Formato de la Microinstrucción : Campos

Campo	Función
Control ALU	Especifica la operación ejecutada en la UAL. El resultado siempre se guardará en ALUOut.
Fuente1	Especifica de dónde viene el primer operando de la UAL.
Fuente2	Especifica de dónde viene el segundo operando de la UAL.
ControlRegistros	Especifica la lectura y escritura del banco de registros, y de dónde viene el valor escrito.
Memoria	Especifica la lectura y escritura de la memoria, y de dónde viene el valor escrito y adónde va el valor leído.
ControlEscrPC (PCWriteControl)	Especifica la escritura en el PC.
Secuenciación	Especifica cómo se elige la siguiente microinstrucción.

Formato de la Microinstrucción : Campos

Campo	Valor simbólico	Señales activas	Comentario
Etiqueta	Cualquier cadena		Se usa para definir etiquetas que clarifican la secuenciación del microcódigo. No generan señales de control, pero se usan entre otras cosas para especificar los contenidos de las tablas de envío (dispatch), cuya selección se resuelve mediante la señal CtrlDir (**AddCtl**).
ControlALU	Add	ALUOp = 00	La UAL suma.
	Subt	ALUOp = 01	La UAL resta y realiza la comparación para verificaciones.
	FuncCode	ALUOp = 10	La UAL realiza la operación indicada por el contenido del campo **Func** de la instrucción.
Fuente1	PC	ALUSrcA = 0	El primer operando de la UAL es el contenido del PC.
	A	ALUSrcA = 1	El primer operando de la UAL es el contenido del registro A.
Fuente2	B	ALUSrcB = 00	El segundo operando de la UAL es el contenido del registro A.
	4	ALUSrcB = 01	El segundo operando de la UAL es la constante 4.
	Extend	ALUSrcB = 10	El segundo operando de la UAL, procede de la salida de la unidad de extensión de signo de 16 bits a 32.
	Extshift	ALUSrcB = 11	El segundo operando de la UAL procede de la salida de la unidad de desplazamiento de dos posiciones hacia la izquierda situada tras la unidad de extensión de signo de 16 bits a 32.

Formato de la Microinstrucción : Campos

Campo	Valor simbólico	Señales activas	Comentario
ControlRegistros	Read		Se leen dos registros usando los campos **rs** y **rt** de la instrucción, y los resultados se guardan en los registros **A** y **B**.
	WriteALU	RegWrite RegDst = 1 MemtoReg = 0	Se escribe un registro usando el campo **rd** de la instrucción con el contenido del registro **ALUOut**.
	WriteMDR	RegWrite RegDst = 0 MemtoReg = 1	Se escribe un registro usando el campo **rt** de la instrucción con el contenido del registro **MDR**.
Memoria	ReadPC	MemRead IorD = 0 IRWrite	Lee de memoria usando la dirección dada por el contenido del **PC**, y escribiendo la información leída en el **IR** (y en el **MDR**).
	ReadALU	MemRead IorD = 1	Lee de memoria usando la dirección dada por el contenido de **ALUOut**, y escribiendo la información leída en el **MDR**.
	WriteALU	MemWrite IorD = 1	Escribe en memoria usando la dirección dada por el contenido de **ALUOut** y el contenido del registro **B** como dato.

Formato de la Microinstrucción : Campos

Campo	Valor simbólico	Señales activas	Comentario
ControlEscrPC	ALU	PCSource = 00 PCWrite	Escribe la salida de la **UAL** en el **PC**.
	ALUOut-Cond	PCSource = 01 PCWriteCond	Si la salida **Zero** de la **UAL** está activa, escribe en el **PC** el contenido del registro **ALUOut**.
	JumpAddress	PCSource = 10 PCWrite	Escribe en el **PC** la dirección de salto de la instrucción.
Secuenciación	Seq	AddCtl = 11	Selecciona la siguiente microinstrucción.
	Fetch	AddCtl = 00	Selecciona la microinstrucción 0 para comenzar la fase de lectura de una nueva instrucción.
	Dispatch1	AddCtl = 01	Selecciona la siguiente microinstrucción mediante la tabla de envío 1.
	Dispatch2	AddCtl = 10	Selecciona la siguiente microinstrucción mediante la tabla de envío 2.

Formato de la Microinstrucción : Tablas de envío

Tabla de envío 1

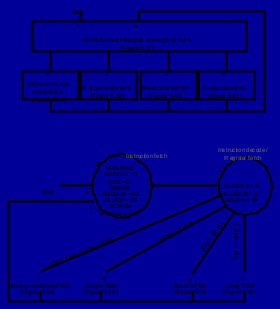
Op[5-0]	Instrucción	Valor simbólico
000000	Instrucción R	Rformat1
000010	Instrucción J	JUMP1
000100	Instrucción BEQ	BEQ1
100011	Instrucción LW	Mem1
101011	Instrucción SW	Mem1

Tabla de envío 2

Op[5-0]	Instrucción	Valor simbólico
100011	Instrucción LW	LW2
101011	Instrucción SW	SW2

Máquina de estados finitos: Etapas comunes

- Representación abstracta
  - Etapas 1 y 2 comunes a todas las instrucciones
- Diagrama de estados
  - Etapas 1 y 2



Creación del microprograma: lectura y decodificación

Acciones:

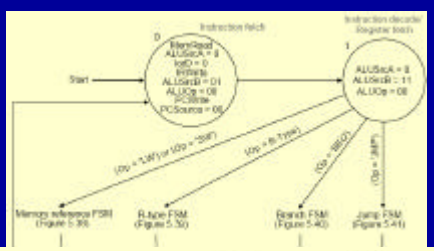
Microinstrucción 0 (instruction fetch):

- Leer de memoria la instrucción y cargarla en IR.
- Incrementar el PC sumándole 4.

Microinstrucción 1:

- Decodificar la instrucción.
- Leer dos registros del banco y guardarlos en A y B.
- Calcular la hipotética dirección de ramificación y guardarla en ALUOut.
- Saltar según tabla de envío 1.

Creación del microprograma: lectura y decodificación



Etiqueta	ControlALU	Fuente1	Fuente2	Control Registros	Memoria	Control EscrPC	Secuenciación
Fetch	Add	PC	4		ReadPC	ALU	Seq
	Add	PC	Extshft	Read			Dispatch1

Creación del microprograma: Ejecución de instrucciones de acceso a memoria

Acciones:

Microinstrucción 2:

- Calcular la dirección del dato en memoria y guardarla en ALUOut.
- Saltar según tabla de envío 2.

Si Op=LW

Microinstrucción 3:

- Leer dato de memoria y guardarlo en MDR (MAR).

Microinstrucción 4:

- Escribir el contenido de MDR en el banco de registros.
- Saltar a microinstrucción 0 (fetch).

Creación del microprograma: Ejecución de instrucciones de acceso a memoria

Si Op=SW

Microinstrucción 5:

•Grabar en memoria el contenido de B en la dirección ALUOut.

•Saltar a microinstrucción 0 (fetch).

Etiqueta	ControlALU	Fuente1	Fuente2	Control Registros	Memoria	Control EscrPC	Secuenciación
Mem1	Add	A	Extend				Despach2
LW2					ReadALU		Seq
				MemWB			Fetch
SW2					WriteALU		Fetch

Creación del microprograma: Ejecución de instrucciones tipo R

Acciones:

Microinstrucción 6:

•Sumar los contenidos de los registros A y B y guardar el resultado en ALUOut.

Microinstrucción 7:

• Escribir el contenido de ALUOut en el banco de registros.

• Saltar a microinstrucción 0 (fetch).

Etiqueta	ControlALU	Fuente1	Fuente2	Control Registros	Memoria	Control EscrPC	Secuenciación
Rformat1	FuncCode	A	B				Seq
				WriteALU			Fetch

Creación del microprograma: Ejecución de instrucción BEQ

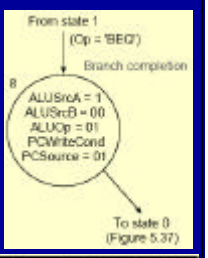
Acciones:

Microinstrucción 8:

• Restar los contenidos de A y B.

• En función de la condición de resultado nulo, escribir en el PC el contenido de ALUOut.

•Saltar a microinstrucción 0 (*fetch*).



Etiqueta	ControlALU	Fuente1	Fuente2	Control Registros	Memoria	Control EscrPC	Secuenciación
BEQ1	Subt	A	B			ALUOut-Cond	Fetch

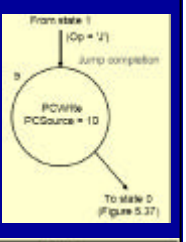
Creación del microprograma: Ejecución de instrucción J.

Acciones:

Microinstrucción 9:

• Escribir en el PC la dirección de salto procedente del desplazador.

• Saltar a microinstrucción 0 (*fetch*).



Etiqueta	ControlALU	Fuente1	Fuente2	Control Registros	Memoria	Control EscrPC	Secuenciación
JUMP1						Jump address	Fetch

Microcódigo completo.

Etiqueta	ControlALU	Fuente1	Fuente2	Control Registros	Memoria	Control EscrPC	Secuenciación
Fetch	Add	PC	4		ReadPC	ALU	Seq
	Add	PC	ExtAlf	Read			Despach1
Mem1	Add	A	Extend				Despach2
LW2					ReadALU		Seq
				WriteMDR			Fetch
SW2					WriteALU		Fetch
Rformat1	FuncCode	A	B				Seq
				WriteALU			Fetch
BEQ1	Sub	A	B			ALUOut Cond	Seq
JUMP1						Jump address	Fetch

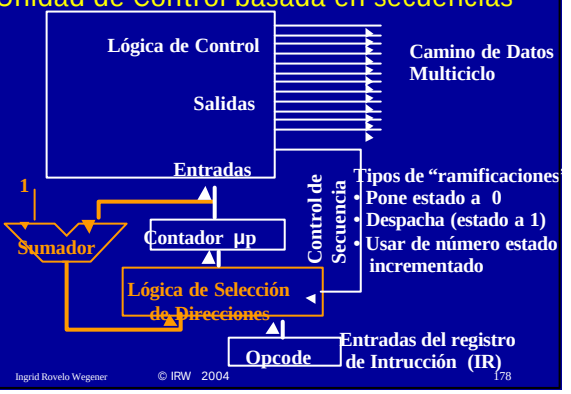
Tabla de envío 1

Op[5-0]	Instrucción	Valor simbólico
000000	Instrucción R	Rformat1
000010	Instrucción J	JUMP1
000100	Instrucción BEQ	BEQ1
100011	Instrucción LW	Mem1
101011	Instrucción SW	Mem1

Tabla de envío 2

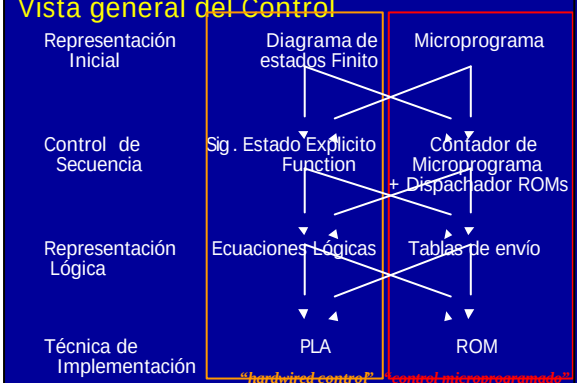
Op[5-0]	Instrucción	Valor simbólico
100011	Instrucción LW	LW2
101011	Instrucción SW	SW2

Unidad de Control basada en secuencias



Tipos de "ramificaciones"
- Pone estado a 0
- Despacha (estado a 1)
- Usar de número estado incrementado

Vista general del Control



Software heredado y la Microprogramación

- Un mismo conjunto de instrucciones para muchas clases de computadoras
  - (8-bit a 64-bit)
- Si la microprogramación pudiera hacer fácilmente el mismo repertorio de instrucciones en diferentes micro arquitecturas , entonces podrían múltiples microprogramas hacer múltiples conjuntos de instrucciones en la misma arquitectura?

Software heredado y la Microprogramación

- "Emulación": interprete del repertorio de instrucciones en micro código para repertorios de instrucciones no nativos.
- Muy exitoso: a inicios de los años de la IBM 360 fue difícil saber cuando un conjunto de instrucciones antiguas o un conjunto de instrucciones nuevas se utilizaba más frecuentemente.

Microprogramación una inspiración para RISC

- Si una sola instrucción se puede ejecutar en una muy alta velocidad del reloj…
- Si se pudiera escribir siempre compiladores para producir microinstrucciones…
- Si la mayoría de los programadores utilizan solo instrucciones y modos de direccionamiento …
- Si el micro código se mantuviera en RAM en vez de ROM así se podrían corregir los errores (bugs) …

Microprogramación una inspiración para RISC

- Si la misma memoria utilizada para el control de la memoria pudiera utilizarse en vez de la caché para las "macroinstrucciones"…
- Entonces porque no saltarnos la interpretación por medio de un microprograma y simplemente compilarlo directamente a un lenguaje de bajo nivel de la máquina? (La microprogramación es poderosísima cuando el ISA iguala el camino de datos 1-1)

Microprogramación Pros y Contras

- Fácil de diseñar el control
- Flexibilidad
  - Fácil de adaptar a los cambios en la organización, tiempo, y tecnología.
  - Se pueden hacer los cambios tardíamente en el diseño del ciclo, o incluso en campo.
- Se puede implementar conjuntos de instrucciones muy poderosas (más que solamente controlar la memoria).
- Generalmente
  - Se pueden implementar un conjunto de instrucciones múltiples en la misma máquina.
  - Se puede dividir el conjunto de instrucciones para alguna aplicación

Microprogramación Pros y Contras

- Compatibilidad
  - Muchas organizaciones, mismo conjunto de instrucciones
- Costoso de implementar
- Lento
- La Microprogramación es un concepto fundamental
  - Implementar un conjunto de instrucciones por medio de la construcción de un procesador muy simple y la interpretación de las instrucciones.
  - Es esencial para instrucciones muy complejas y cuando son posibles pocas transferencias de registros.

Microinstrucciones

• La microprogramación fue inventada por Maurice V. Wilkes en 1953.

• En aquellos tiempos la tecnología no estaba suficientemente adelantada como para contar con una memoria de control suficientemente rápida.

• IBM
  • rescató la microprogramación en 1964 con su familia de computadores 360.

  •incorporó el desarrollo de tecnología para memorias en la compañía para hacer viable la microprogramación.

Microinstrucciones

• Las primeras máquinas microprogramadas implementaban el control mediante una memoria ROM: de ahí la terminología utilizada.

• **Microinstrucción**: conjunto de valores binarios de las señales de control.

• "Su función, estructura y misión es análoga a la de las instrucciones en código máquina (representación binaria de las operaciones que puede realizar un computador).

• **Microprograma**: secuencia de microinstrucciones necesarias para ejecutar las instrucciones.
"Su función es análoga a la de los programas en código máquina (secuencia de instrucciones necesarias para ejecutar los algoritmos).-

Microinstrucciones

•**Micromemoria**: memoria de control en ROM que contiene las microinstrucciones.

  •"Su función es análoga a la de la memoria central en RAM.

• **Contador de microprograma**: contiene la microdirección (dirección en la micromemoria) de la siguiente microinstrucción del microprograma y recorre el mismo en su ejecución.

  • "Su función y nombre son análogos a los del contador de programa (contiene la dirección en memoria RAM de la siguiente instrucción del programa y recorre el mismo en su ejecución).

Microinstrucciones

La microprogramación permite modificar y depurar fácilmente el control.

- El uso de micromemorias RAM facilitó la depuración incluso con la máquina ya en el mercado.
- La microprogramación abarata el costo de ampliar el repertorio de instrucciones para incluir instrucciones nuevas, complejas o cuya ejecución implique muchos ciclos de reloj.
- Basta con incluir una memoria de control suficientemente grande.
- Si la memoria de control tenía parte RAM el usuario incluso podía definir nuevas instrucciones.

Microinstrucciones

- La microprogramación puso de moda la emulación: simulación completa del repertorio de instrucciones de una máquina gracias a un control microprogramado.
- La microprogramación estuvo muy de moda en los años 60 y 70.
- Las UCP se diseñaban mediante componentes discretos MSI.
- Alternativas: control cableado frente a control microprogramado (***firmware***: microcódigo en ROM o incluso RAM).
- Hoy en día el diseño de UCP con unidad de control microprogramada está en desuso.

Microinstrucciones

- El tiempo de acceso a la memoria de control a la RAM de instrucciones es similar: la solución microprogramada es muy lenta.
- La UCP se realiza en un solo chip (microprocesador), no mediante componentes discretos.
- Están en boga los diseños RISC, que implican un control más sencillo.
- En cualquier caso, para realizar el control se recurre a herramientas CAD.

Microinstruciones

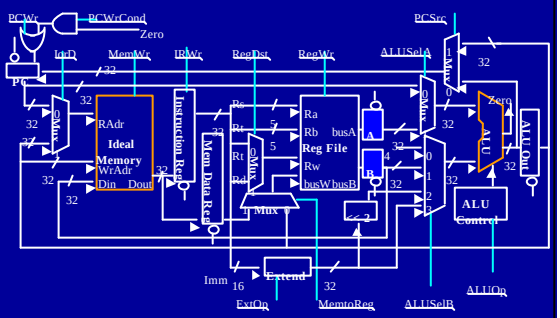
La microprogramación como técnica para especificar el control simplifica el proceso de diseño, ya que:

- Permite especificar el control de forma simbólica.
- Permite comprender más fácilmente el funcionamiento de la máquina.
- Facilita la identificación de los patrones de activación de las señales de control, así como su secuenciamiento .
- Una vez especificado el microcódigo, puede recurrirse a un diseño microprogramado

o bien puede diseñarse un control con un secuenciador para especificar la función de transición.

Ambas opciones son equivalentes.

Camino de datos Multiciclo



Máquina de estados finitos: Diagrama completo

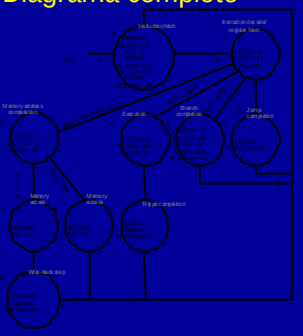


Diagrama de estados multiciclo

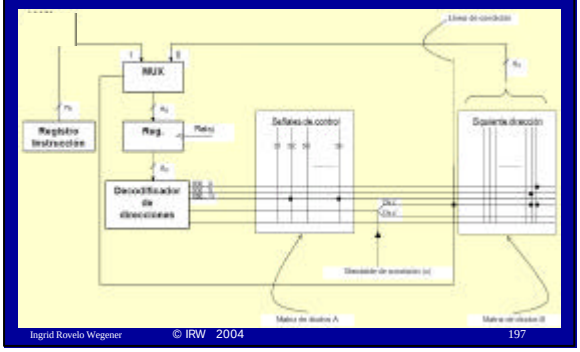
El microcódigo puede residir en una ROM (llamada memoria de control).

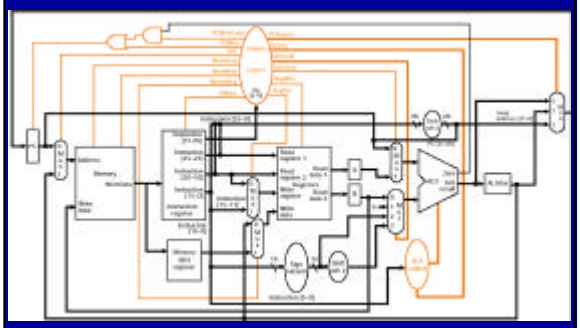
Las ROM son más flexibles.

El contador de microprograma viene a ser un registro de estado.

Las señales de control forman la palabra de control, que a veces se almacena en un registro denominado registro de control.

El primer diseño de unidad de control microprogramada se debe a Maurice V. Wilkes.





Introducción a las aplicaciones

En una máquina con repertorio de instrucciones complejo existen muchos factores que complican el control:

- Amplia variedad de modos de direccionamiento.
- Muchas instrucciones.
- Muchas variantes de las instrucciones en función de los modos de direccionamiento de los operandos.
- Instrucciones con diferentes formatos y longitudes.

Aplicaciones

Cuando el control es complicado, la técnica de diseño de la unidad de control mediante un diagrama de estados es inaplicable.

Por todo ello, en el proceso de diseño de la unidad de control de un computador se hace necesario contar con técnicas o herramientas que simplifiquen la tarea.

- La circuitería se suele diseñar con la ayuda de herramientas CAD.
- Microprogramación: técnica que permite especificar y diseñar el control de forma análoga a como se realiza un programa de computador.

Aplicaciones

- Si crece el repertorio de instrucciones, también crece el número de estados, y con él la lógica de control, especialmente la dedicada a la función de transición.
- Si hay instrucciones que duran muchos ciclos de reloj, en su ejecución se producen largas secuencias de estados con un solo camino que los une.
- Se producen muchísimas combinaciones estado+IR imposibles.

Aplicaciones

- Sin embargo, gran parte de la lógica se dedica a la función de transición.
- Cuando el control es complicado, la técnica de diseño de la unidad de control mediante un diagrama de estados es inaplicable.
- El microprograma es un intérprete, busca las instrucciones de lenguaje de máquina tales como ADD, MOVE, JUMP

Aplicaciones

Cuando el control es complicado, la técnica de diseño de la unidad de control mediante un diagrama de estados es inaplicable.

Por todo ello, en el proceso de diseño de la unidad de control de un computador se hace necesario contar con técnicas o herramientas que simplifiquen la tarea.

- La circuitería se suele diseñar con la ayuda de herramientas CAD.
- Microprogramación: técnica que permite especificar y diseñar el control de forma análoga a como se realiza un programa de computador.

Aplicaciones de la Microprogramación

- Realización de Computadores
- Emulación
- Soporte de Sistemas Operativos
- Soporte de lenguajes de alto nivel

Aplicaciones de la Microprogramación

- Microdiagnóstico
- Adaptación al usuario
- Realización de nuevas instrucciones
- La microprogramación se usa actualmente en periféricos (módems, impresoras, etc)