Arquitectura de Computadoras

Antes se tenían sistemas cableados:

* Programación en hardware: Cuando cambiamos las tareas, debemos cambiar el hardware.
  + Datos 🡺 Secuencia de funciones aritmético/lógicas 🡺 Resultados

Ahora:

* Programación en software: En cada paso se efectúa alguna operación sobre los datos.

Código de instrucción

||

||

\/

Intérprete de instrucción

|  |
| --- |
| Señales  de  control |

\/

Datos 🡺 Funciones aritmético/lógicas 🡺 Resultados

* + Tengo instrucciones que deben ser interpretadas, por cada una de ellas tengo señales de control, las cuales se encargan de administrar los eventos que van a ocurrir.
  + Las instrucciones proporcionan esas señales de control.

Arquitectura de Von Neumann:

* La unidad central de procesamiento (CPU) está constituida por la unidad de control (UC) y la unidad aritmético-lógica (ALU).
* Datos e instrucciones deben introducirse en el sistema y los resultados se. Proporcionarán mediante componentes de E/S.
* Se necesita almacenar temporalmente datos e instrucciones:
  + Memoria Principal

Repertorio de Instrucciones:

* Es el conjunto completo de instrucciones que se realizan en una CPU:
  + Código máquina
  + Binario
* Representado simbólicamente por un conjunto de código de ensamblaje:
  + De operaciones: ADD, SUB, LOAD
  + De operandos: ADD BX, VAR;

Elementos de una instrucción:

1. Código de operación (“Cod op”)
2. Referencia a operandos fuentes
3. Referencia al operando resultado
4. Referencia a la siguiente instrucción (PC) 🡪 Program Counter

¿Dónde se almacenan los operandos?

* Memoria principal (o virtual o en memoria cache)
* Registro de la CPU
* Dispositivos de E/S

Tipos de instrucciones:

* Procesamiento de datos 🡪 Instrucciones aritmético-lógicas
* Almacenamiento de datos 🡪 Instrucciones de memoria
* Transferencia de datos 🡪 Instrucciones de E/S
* Control 🡪 Instrucciones de testeo y flujo del programa

Cantidad de direcciones:

* + - Más direcciones por instrucción
      * Instrucciones más complejas
      * Más registros: las operaciones entre registros son más rápidas
      * Menos instrucciones por programa
    - Menos direcciones por instrucción
      * Instrucciones menos complejas
      * Más instrucciones por programa
      * La captación/ejecución de las instrucciones es más rápida

Decisiones en el diseño del conjunto de instrucciones:

* Tipos de operandos (datos)
* Repertorio de operaciones:
  + Cuántas operaciones se consideran
  + Cuáles operacions se realizan
  + Cuán complejas será cada una de ellas
* Formatos de instrucciones:
  + Longitud de instrucción
  + Número de direcciones
  + Tamaño de los campos
* Registros
  + Número de registros de la CPU que se pueden referenciar
  + En qué registros se pueden ejecutar qué operaciones
* Modos de direccionamiento
* RISC contrapuesto a CISC (Computadores de conjunto reducido de instrucciones) a (Computadoras de conjunto complejo de instrucciones).

Tipos de operandos:

1. Direcciones
2. Números (punto fijo o punto flotante)
3. Caracteres
4. Datos lógicos (Bits 0s o 1s)

Orden de los bytes:

* Big endian: Se almacena el byte más significativo en la dirección con valor numérico más bajo.
* Little endian: Se almacena el byte menos significativo en la dirección con valor numérico más bajo.

Transferencia de datos:

* Debe especificarse:
  + Ubicación del operando fuente
  + Ubicación del operando destino
  + Tamaño de los datos a ser transferidos
  + Modo de direccionamiento
* Diferentes movimientos 🡪 Diferentes instrucciones
  + - Reg-Reg, Reg-Mem, Mem-Reg
* O una instrucción y diferentes direcciones
  + - MOV destino, fuente; copia fuente a destino

Control de flujo 🡪 Modifican el valor contenido en el registro PC

* Salto Incondicional (JMP pos)
* Salto Condicional (ej.: JZ pos)
* Salto con retorno o llamada a subrutina (CALL subrut)

Modos de Direccionamiento:

1. Inmediato 🡪 en la instrucción tengo el operando a usar
2. Directo “de memoria” o Absoluto 🡪 tengo la dirección del operando
3. “Directo” de Registro 🡪 tengo el registro donde se almacena el operando
4. Indirecto “en memoria” 🡪
5. Indirecto “con registro” 🡪
6. Indirecto con Desplazamiento 🡪 tengo el desplazamiento y el registro donde se encuentra la dirección (se suman para obtener la dirección efectiva del operando)
   1. Basado, indexado o relativo al PC
   2. Pila (o relativo al SP)

Ciclo de captación:

* + - La dirección de la instrucción que se debe captar se encuentra en el registro Contador de Programa (PC).
    - La UC capta la instrucción desde la Memoria.
      * La instrucción va al registro de instrucción (IR)
    - El registro PC se incrementa (a no ser que se indique lo contrario).
    - La UC interpreta la instrucción captada y debe llevar a cabo la acción requerida.

Subrutinas:

* Programa auto-contenido
* Puede invocarse desde cualquier punto de programa (mediante la instrucción CALL)
* Brinda economía (reutilización de código) y modularidad (subdividir en pequeñas unidades)
* Requiere pasaje de argumentos (parámetros): por valor (copia) o por referencia (dirección)

Pasaje de argumentos:

1. Vía Registros:
   1. El número de registros es la principal limitación
   2. Es importante documentar qué registros se usan
2. Vía Memoria:
   1. Se usa un área definida de memoria (RAM)
   2. Difícil de estandarizar
3. Vía Pila (Stack):
   1. Es el método más ampliamente usado
   2. El verdadero “pasaje de parámetros”
   3. Independiente de memoria y registros
   4. Hay que comprender bien cómo funciona porque es usada tanto por el usuario como por el sistema

Funcionamiento de una pila:

* + - El operando está en la cabeza de la pila
    - Se requiere un registro Puntero de Pila (SP)
    - Operaciones sobre la pila
      * PUSH; operación de Apilar
      * POP; operación de Desapilar
      * Secuencia de dos acciones:
        + Movimiento de datos Reg-Mem o Mem-Reg
        + Modificación del puntero antes/después de la anterior

Posibles pasos en un procedimiento:

1. Salvar el estado de BP
2. Salvar el estado de SP (BP = SP)
   1. Establece a BP como puntero de referencia y es usado para acceder a los parámetros y datos locales en la pila. SP no puede ser usado para éste propósito porque no es un registro baso o índica. El valor de SP puede cambiar pero BP permanece ‘quieto’.
   2. BP es el puntero al área de la pila asignada al procedimiento (frame pointer).
   3. Para acceder a los datos se deberá sumar un desplazamiento fijo a BP.
3. Reservar espacio para datos locales (opcional)
   1. Se decrementa SP, reservando lugar en la pila
   2. El sistema puede utilizar al SP sin escribir sobre el área de trabajo del procedimiento.
4. Salvar valores de otros registros (opcional)
   1. Si el procedimiento no cambia el valor de los registros, estos no necesitan ser salvados. Normalmente los registros son salvados después de establecer el puntero y los datos locales.
5. Acceder a parámetros
   1. En general el desplazamiento de BP para acceder a un parámetro es igual a:
      1. 2 (tamaño de BP apilado) + tamaño de dirección de retorno + total de tamaño de parámetros entre el buscado y BP.
6. Escribir sentencias a ejecutar
7. Retornar parámetros (opcional)
8. Regresar correctamente del procedimiento
   1. Los registros salvados en la pila deben ser descargados en oren inverso.
   2. Si se reservó espacio para variables locales, se debe reponer SP con el valor de BP que no cambió durante el procedimiento.
   3. Reponer BP.
   4. Volver al programa que llamó al procedimiento con RET.

Anidamiento de subrutinas 🡪 Una subrutina puede llamar a otra

Interrupciones 🡪 Mecanismo mediante el cual se puede interrumpir el procesamiento normal de la CPU.

* Pueden ser de origen interno o externo a la CPU.
* Por qué interrumpir
  + Por resultado de una ejecución de una instrucción (overflow, división por cero)
  + Por un temporizador interno del procesador
  + Por una operación de E/S (para indicar la finalización normal de una operación)
  + Por un fallo de hardware (error de paridad de memoria, pérdida de energía)
* Qué hacer:
  + En casi todos los casos, implica transferir el control a otro programa (GESTOR) que:
    - Salve el estado del procesador
    - Corrija (o responda a) la causa que ocasionó la interrupción
    - Restaure el estado original del procesador
    - Retorne a la ejecución normal del programa interrumpido

Jerarquía de interrupciones 🡪 Si hay múltiples fuentes que pueden solicitar interrupción se establece cuáles son más importantes. Se consideran:

* + - No enmascarables: Las que NO pueden ignorarse 🡪 Indican eventos peligrosos o de alta prioridad.
    - Enmascarables: Pueden ser ignoradas 🡪 Con instrucciones podemos inhibir la posible solicitud.
* Clasificación:
  + Interrupciones por Hardware (las “verdaderas” interrupciones / Interrupt Request) 🡪 Son las generadas por dispositivos de E/S
    - * El sistema de cómputo tiene que manejar estos eventos externos “no planeado” o “asincrónicos”.
      * No están relacionadas con el proceso en ejecución en ese momento
  + Traps/Excepciones 🡪 Interrupciones por hardware creadas por el procesador en respuesta a ciertos eventos como:
    - * Condiciones excepcionales: overflow en ALU de punto flotante.
      * Falla de programa: tratar de ejecutar una instrucción no definida.
      * Fallas de hardware: error de paridad de memoria.
      * Accesos no alineados o a zonas de memoria protegidos.
  + Interrupciones por Software 🡪 Instrucciones explícitas que afectan al procesador de la misma manera que las interrupciones por hardware:
    - * Generalmente usadas para hacer llamadas a funciones del SO (las subrutinas del sistema se cargan en cualquier lugar).
      * No requieren conocer la dirección de la rutina en tiempo de ejecución.
      * Hay sistemas que NO permiten hacer una llamada directa a una dirección de la función del SO, por estar en una zona reservada.

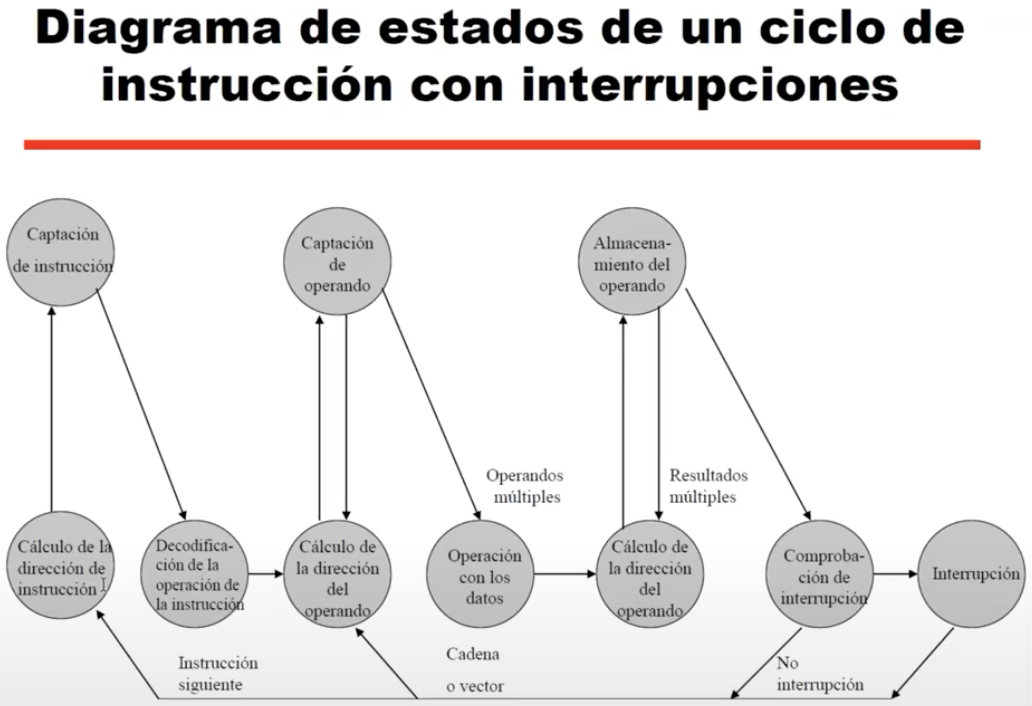
\*Qué pasa si no tuviera estas interrupciones?

- Debería escribir todas las funciones que necesito, o

- Al cargar un programa habría que “mirar” todas las llamadas a funciones del BIOS y SO, y reemplazar en el código las direcciones de todas las funciones invocadas.

Ciclo de Instrucción:

1. Captación 🡪 Captar la siguiente instrucción
2. Ejecución 🡪 Ejecutar la instrucción
3. Gestión de interrupciones 🡪 Si hay pedido de interrupción, interrumpir el proceso
   1. Se comprueba si se ha solicitado alguna interrupción (indicada mediante un flag)
   2. Si no hay señal se capta la siguiente instrucción
   3. Si hay algún pedido de interrupción pendiente:
      1. Se suspende la ejecución del programa en curso
      2. Guarda su contexto (próxima instrucción y el estado del procesador)
      3. Carga el PC con la dirección de comienzo de una rutina de gestión de interrupción. Se inhiben otras interrupciones
      4. Finalizada la rutina de gestión, el procesador retoma la ejecución del programa del usuario en el punto de interrupción (con la instrucción IRET)



Interrupciones Múltiples:

* Interrupciones inhabilitadas:
  + El procesador puede y debe ignorar la señal de petición de interrupción si se produce una interrupción en ese momento.
  + Si se hubiera generado una interrupción se mantiene pendiente y se examinará luego una vez que se hayan habilitado nuevamente.
  + Ocurre una interrupción, se inhabilitan, se gestiona la misma y luego se habilitan otra vez.
  + Por lo tanto las interrupciones se manejan en un orden secuencial estricto.
* Definir prioridades:
  + Una interrupción de prioridad más alta puede interrumpir a un gestor de interrupción de prioridad menor.
  + Cuando se ha gestionado la interrupción de prioridad más alta, el procesador vuelve a las interrupciones previas (de menor prioridad).
  + Terminadas todas las rutinas de gestión de interrupciones se retoma el programa del usuario.

Reconocimiento de interrupciones:

* + - Interrupciones multinivel
      * Cada dispositivo que puede provocar interrupción tiene una entrada física de interrupción conectada a la CPU.
      * Es muy sencillo, pero muy caro.
    - Línea de interrupción única
      * Una sola entrada física de pedido de interrupción a la que están conectados todos los dispositivos.
      * Se debe “preguntar” a cada dispositivo si ha producido el pedido de interrupción (técnica Polling/encuesta).
    - Interrupciones vectorizadas
      * El dispositivo que quiere interrumpir además de la señal de pedido de interrupción, debe colocar en el bus de datos un identificador (vector).
        + Lo coloca el periférico directamente, o
        + Controlador de Interrupciones (que se ocupa de todo).

PIC 🡺 Dispositivo Controlador Programable de Interrupciones