**Clase 1 A – Repaso de Conceptos**

**Arquitectura de Von Neumann**

* CPU 🡪 Unidad Central de Procesamiento
  + UC 🡪 Unidad de Control
  + ALU 🡪 Unidad Aritmético-Lógica
* 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
* Registros de la CPU
  + PC 🡪 Contador de Programa. Próxima dirección a acceder.
  + IR 🡪 Registro de Instrucción de la unidad de control, por donde pasa la instrucción una vez que fue captada.
  + MAR 🡪 Registro de dirección de memoria (interconexión de los buses)
  + MBR 🡪 Registro de buffer de memoria (interconexión de los buses)
  + E/S AR 🡪 Registro de dirección de E/S
  + E/S BR 🡪 Registro buffer de E/S

**Instrucciones**

Elementos:

* Código de operación (“Cod Op”)
* Referencia a operandos fuentes
* Referencia al operando resultado
* Referencia a la siguiente instrucción (PC)

¿Dónde se almacenan los operandos?

* Memoria Principal (o virtual o cache)
* Registros de la CPU
* Dispositivo de E/S

Tipos:

* 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 de programa

Cantidad de direcciones:

|  |  |
| --- | --- |
| Más direcciones por instrucción | Instrucciones más complejas |
| Más registros (operaciones 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 |

RISC vs. CISC (Computadora de conjunto reducido/complejo de instrucciones)

Big Endian 🡪 El byte más significativo en la dirección con valor numérico más bajo

Little Endian 🡪 El byte menos significativo en la dirección con valor numérico más bajo

Modos de Direccionamiento:

1. Inmediato: El operando está en la instrucción
2. Directo de memoria o Absoluto: En la instrucción está la dirección del operando
3. Directo de Registro: La instrucción tiene el registro que guarda al operando
4. Indirecto de memoria
5. Indirecto con registro
6. Indirecto con Desplazamiento: La instrucción tiene el desplazamiento y el registro donde está la dirección del operando, debo sumar los valores para obtener la dirección efectiva
   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 (va al IR)
* El registro PC se incrementa
* La UC interpreta la instrucción captada y debe llevar a cabo la acción requerida

**Clase 1 B – Pasaje de parámetros a subrutina**

**Subrutina**

* Programa auto-contenido
* Puede invocarse desde cualquier punto de un programa (instrucción CALL)
* Brinda economía (reúso de código) y modularidad (subdividir en partes)
* Requiere pasaje de argumentos (parámetros)
  + Por valor (copia de una variable)
  + Por referencia (dirección de la variable)

**Pasaje de parámetros**

* Vía Registros
  + El número de registros es la principal limitación
  + Es importante documentar qué registros se usan
* Vía Memoria
  + Se usa un área definida de memoria (RAM)
  + Difícil de estandarizar
* Vía pila (stack)
  + Es el método más ampliamente usado
  + El verdadero “pasaje de parámetros”
  + Independiente de memoria y registros
  + Hay que comprender bien cómo funciona porque la pila es usada por el usuario y por el sistema

**Pila**

* El operando está (implícitamente) en la cabeza de la pila
* Se requiere un registro Puntero de Pila (SP)
  + Contiene la dirección de la cabeza de la pila
* Operaciones sobre la pila:
  + PUSH 🡪 Apilar
  + POP 🡪 Desapilar

Posibles pasos en un procedimiento:

1. Salvar el estado de BP (viejo BP) 🡪 push BP
2. Salvar el estado de SP (BP=SP) 🡪 mov BP, SP
3. Reservar espacio para datos locales (opcional)
4. Salvar valores de otros registros (opcional)
5. Acceder a parámetros
   1. Desplazamiento de BP = 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

Salida del procedimiento:

1. Los registros salvados en la pila deben ser descargados en orden 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 llama a otra dentro de ella

Debo repetir el procedimiento de llamado a la subrutina

**Clase 2 – Interrupciones**

**Interrupciones**

🡺 Mecanismo mediante el cual se puede interrumpir el procesamiento normal de la CPU

🡪 Pueden ser de origen interno o externo a la CPU

Motivos:

* Por resultado de una ejecución de una instrucción (Overflow, div por 0)
* Por un temporizador interno del procesador
* Por una operación de E/S (uso de un periférico)
* Por un fallo de hardware (error de paridad en memoria, pérdida de energía)

Gestor 🡪 código que atiende al pedido de interrupción

* Se transfiere el control al gestor:
  + Salva el estado del procesador
  + Corrige (o responde a) la causa que ocasionó la interrupción
  + Restaura el estado original del procesador
  + Retorna a la ejecución normal del programa interrumpido

¿El gestor salva el estado del procesador o se debe hacer antes de darle el control?

Jerarquía:

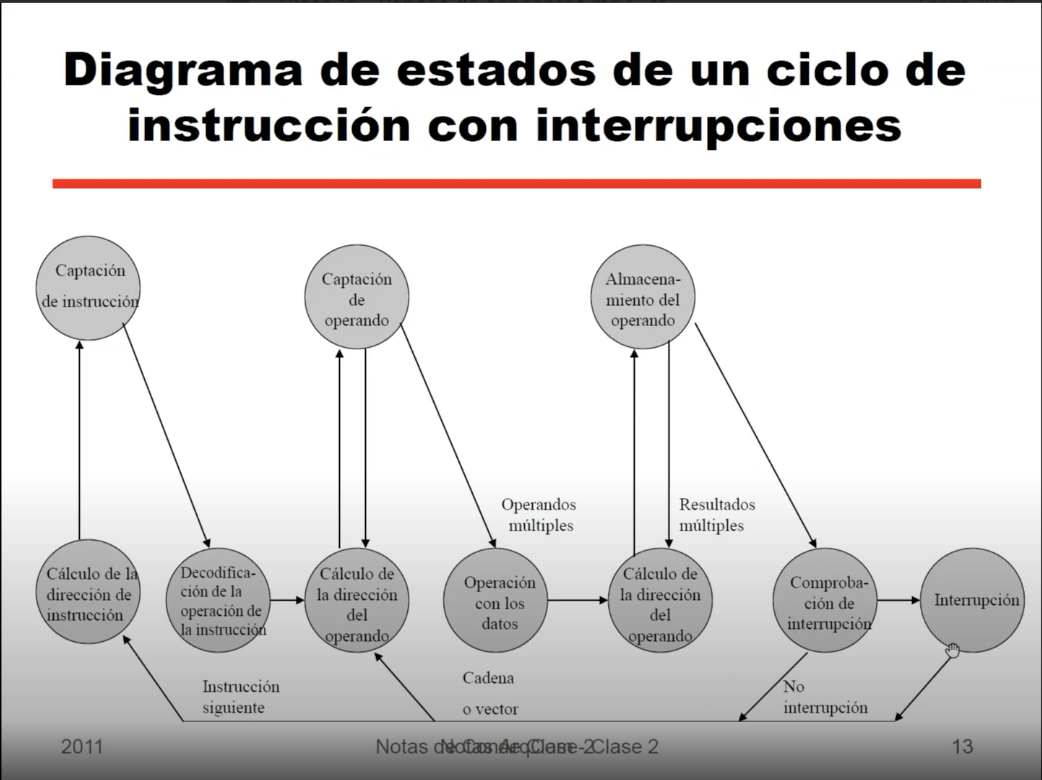
* No Enmascarables 🡪 Las que NO pueden ignorarse
* Enmascarables 🡪 Pueden ser ignoradas (con instrucciones)

Tipos:

* Por Hardware
  + Son las generadas por dispositivos de E/S (periféricos)
  + Son las VERDADERAS interrupciones
  + El sistema de cómputo tiene que manejas estos eventos externos “no planeados” o asincrónicos”
  + No están relacionadas con el proceso en ejecución en ese momento
  + Son conocidas como interrupt request
* Traps/Excepciones 🡪 Interrupciones por hardware creadas por el procesador en respuesta a ciertos eventos:
  + 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 a zonas de memoria protegidos
* Por Software
  + Generalmente usadas para hacer llamadas a funciones del SO
    - Esto permite que las subrutinas del sistema se carguen en cualquier lugar
  + No requieren conocer la dirección de la rutina en tiempo de ejecución

Ciclo de Interrupción

1. Se comprueba si se ha solicitado alguna interrupción (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 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 (instrucción IRET), el procesador retoma la ejecución del programa del usuario en el punto de interrupción



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 🡪 Para pasar el control al gestor

* Interrupciones multinivel
  + Cada dispositivo que puede provocar interrupción tiene una entrada física de interrupción conectada a la CPU
  + Es muy sencillo, pero muy caro (dos conexiones, uno de pedido y otro de confirmación)
* 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 su identificador (vector)
    - Lo coloca el periférico directamente, o
    - Controlador de Interrupciones (que se ocupa de todo)

PIC 🡺 Dispositivo Controlador Programable de Interrupciones

* Es el encargado de controlar todos los pedidos de interrupciones
* Estarán conectados los periféricos a través de sus módulos de E/S

**Módulo de E/S**

**Técnica de movimiento de datos de/hacia periféricos**

**Segmentación de cause**

**Arquitectura RISC**

**Memoria Caché**

**Arquitectura Superescalares**

**Taxonomía de Flynn**

**Sistemas paralelos**