AC Primer Parcial

1. Fundamentos de diseño y evaluación de computadores

1.1 Conceptos básicos:

Tenerlos claros y poco mas, no hay preguntas teóricas

- **Rendimiento**: Es la **inversa del tiempo** que tarda en completarse una tarea, se puede mejorar con las siguientes optimizaciones:
    - Jerarquía de Memoria
    - Paralelismo
    - Segmentación
- **Consumo**: Energía consumida por unidad de tiempo (watios).
- **Fiabilidad**: Tiempo entre fallos/reparaciones. Sistemas tolerantes a fallos.
- **Factor de yield**: Fracción de circuitos correctos.
- **Latencia**: Tiempo que transcurre entre la solicitud de un dato y la disponibilidad del mismo. (Se mide en ciclos o unidades de tiempo)
- **Ancho de banda**: Número de bytes transmitidos por unidd de tiempo.

1.2 Evaluación del coste

No muy importante

- Coste de un circuito integrado:

$$Coste\ citcuito = \frac{Coste\ die + Coste\ testeo + Coste\ empaquetado}{Yield\ final(test)}$$

- Coste del die (dado):

$$Coste\ die = \frac{Coste\ waffer}{Dies\ per\ waffer \cdot\ Die\ yield}$$

- Dies per waffer (oblea):

$$Dies\ per\ waffer = \frac{Area\ útil}{Die\ area} = \frac{\pi \cdot (diametro/2)^2}{Die\ area} - \frac{\pi \cdot diamenter}{\sqrt{2 \cdot Die\ area}}$$

- Die yield

$$Die\ yield = Waffer\ yield \cdot \frac{1}{(1 + defectos\ por\ unidad\ area \cdot\ die\ area)^{\alpha}}$$

- α = media de la complejidad, se aproxima al número de máscaras críticas

1.3 Rendimiento de un procesador

SUPER IMPORTANTE

Tiempo ejecución

$$1/Rendimiento = Tiempo\ de\ ejecución = N \cdot CPI \cdot Tc$$

Donde:

N: Es el número de instrucciónes que se ejecutan, instrucciones dinámicas.

CPI: Es el número medio de ciclos por instrucción.

Tc: Es el tiempo de ciclo.

Métricas de Rendimiento:

MIPS: Millones de **instrucciones** por segundo.

MFLOPS: Millones de **operaciones en punto flotante** por segundo.

Ganancia (o speedup)

$$Ganancia = Ta/Tb$$

- Si > 1 entonces B és más rápido que A
- SI < B es més lento que A

En porcentajes:

$$Ganancia(porcentaje) = (Ta/Tb\ - 1) \cdot 100$$

Ley de Amdahl

Asumimos que Fm es la parte del programa que no se puede optimizar, así que asumimos al infinito (el resto del programa que si se puede optimitzar vale tarda 0).

$$Ganancia = 1/(1 - Fm)$$

1.4 Consumo

Muy importante

Potencia

$$Potencia\ (W) = I \cdot V$$

Energía

$$Energía\ (J) = P \cdot T = I \cdot V \cdot T$$

Potencia y energía de conmutación

$$Potencia = C \cdot V^2 \cdot f$$

$$Energía = C \cdot V^2$$

Potencia de fugas

$$Potencia = Ifuga \cdot V$$

Potencia consumida CMOS

$$Pcons = Pconmutación + Potencia\ Fugas + Corriente\ Cortocircuito$$

Métricas de eficiencia

$$Eficiencia\ energética = \frac{rendimiento}{watio} = \frac{1}{tiempo \cdot watio} = \frac{1}{energíaconsumida}$$

1.5 Fiabilidad

Bastante importante

Fiabilidad: Tiempo de funcionamiento continuo sin fallos

Tasa de fallos:

- $\lambda = \frac{1}{MTTF}$

Interrupción del servicio

- $MTTR = Mean\ Time\ To\ Repair$

Tiempo medio entre fallos

- $MTBF = MTTF + MTTR$

Disponibilidad

- $Availability = \frac{MTTF}{MTTF+MTTR}$

Probabilidad que se poduzca un fallo

- $p = 1 - e^{-\lambda t}$

2. Interfaz Alto Nivel - Ensamblador

Diapos: 7, 9, 14, 18 - 22

2.2 Structs

- Es representen d'adalt abaix, +0 fins al final.
- Els structs s'alinean al seu element que tingui una restricció d'alineament mes alta.

Alineaments:

- char: 1-byte
- short: 2-bytes
- int: 4-bytes
- puntero: 4-bytes
- double: 8-bytes
- Long double: 12-bytes

2.3 Subrutinas

- Bloque de activacion "hacia arriba" los parametros que se guarden
- En medio el %ebp que es el 0 y despues @RET que es el +4
- Bloque de activación "hacia abajo" los parametros que pasan a la funcion

1. Paso de parámetros: pushl OPERANDO
    - Donde operando puede ser Memoria, Registro o literal
    - Ejemplo: pushl %eax
2. Llamada subrutina: call etiqueta | call OPERANDO
    - Donde operando puede ser Memoria o registro.
    - Ejemplos: call for | call (%ebx)
3. Puntero bloque de activación: Se tiene que gaurdar siempre el %ebp

    ```
    pushl %ebp
    movl %esp, %ebp
    subl $4, %esp
    ```

4. Reserva espacio variables locales: Se tiene que reservar el espacio a usar en 5
    - Ejemplo: subl $8, %esp
5. Salvar estado llamador: Se tiene que hacer si usamos %ebx, %esi, %edi
    - Para guardar un vector se tienen que guardar **todos** los elementos del vector
    - Ejemplo: pushl %ebx
6. Cuerpo subrutina
7. Mover resultado a eax: El resultado que queremos hacer return lo ponemos en eax
    - Ejemplo: movl -4(%ebp), %eax
8. Restaurar estado: Hacer popl de todos los que hemos guardado en 5
    - Ejemplo: popl %ebx
9. Elimina variables locales: Restaurar el %esp
    - Ejemplo: movl %ebp, %esp
10. Deshacer enlace dinámico: popl %ebp
11. Retorno de subrutina: ret

12. Elimina parámetros: Hacer addl de todo el bloque de activación
13. Recoger/usar resultado: Recordar que esta guardado en %eax

3. Jerarquía de Memorias

Algorithmos de emplazamiento

Directo: TAG + #Linea MC + #bytes

- Linea MC = Bloque de memoria mod #lineas MC

Asociativo por conjuntos: TAG + #Conjuntos MC + #bytes

- Conjunto MC = Bloque de memoria mod #conjuntos

Completamente asociativo:

Algoritmos de reemplazo

Reemplazo Aleatorio:

- Funciona mejor de lo esperado
- Muy sencillo de implementar

Reemplazo FIFO:

- Comportamiento no deseado patológico
- Muy sencillo de implementar

Reemplazo LRU:

- Da buenos resultados
- Muy costoso de implementar
- Se suele cambiar por un PseudoLRU

Políticas de Escritura

ACTUALIZACION DE LA MEMORIA PRINCIPAL

Write through: (escritura a través o escritura inmediata)

- Se escribe **simultaneamente a MC y a MP**

- Tiempo de servicio = tiempo acceso MP
- Se puede reducir el tiempo usando buffers
- MP siempre actualizada

Copy Back: (escritura diferida)

- **Solo se escribe MC**
- Hay un Dirty Bit que controla si esa linea ha sido modificada
- Se actualiza el bloque cuando se va a reemplazar
- Menos tiempo escritura, Mas tiempo de fallo
- Hay inconsistencia

QUE HACER EN CASO DE FALLO DE ESCRITURA

Write allocate: (migración en caso de fallo)

- Se trae el bloque de MP a MC y después se raliza la escritura

Write no allocate: (sin migración en caso de fallo)

- El bloque **NO** se trae a MC. Obliga ha hacer escritura directamente a MP.

Calcular TMA

$$TMA = h \cdot tsa + m \cdot tsf = tsa + m \cdot tpf$$

Donde:

- Tsa: Coste acierto
- Tsf: Coste fallo $tsf = tsa + tpf$
- Tpf: Penalización de un fallo

Calcular CPImem

$$CPImem = nr \cdot (Tma - tsa)$$

Donde:

- nr: Numero medio de referencias por instrucción