laSalle ENG

Universitat Ramon Llull

Sistemes Operatius

Capítol 1: Introducció als Sistemes Operatius

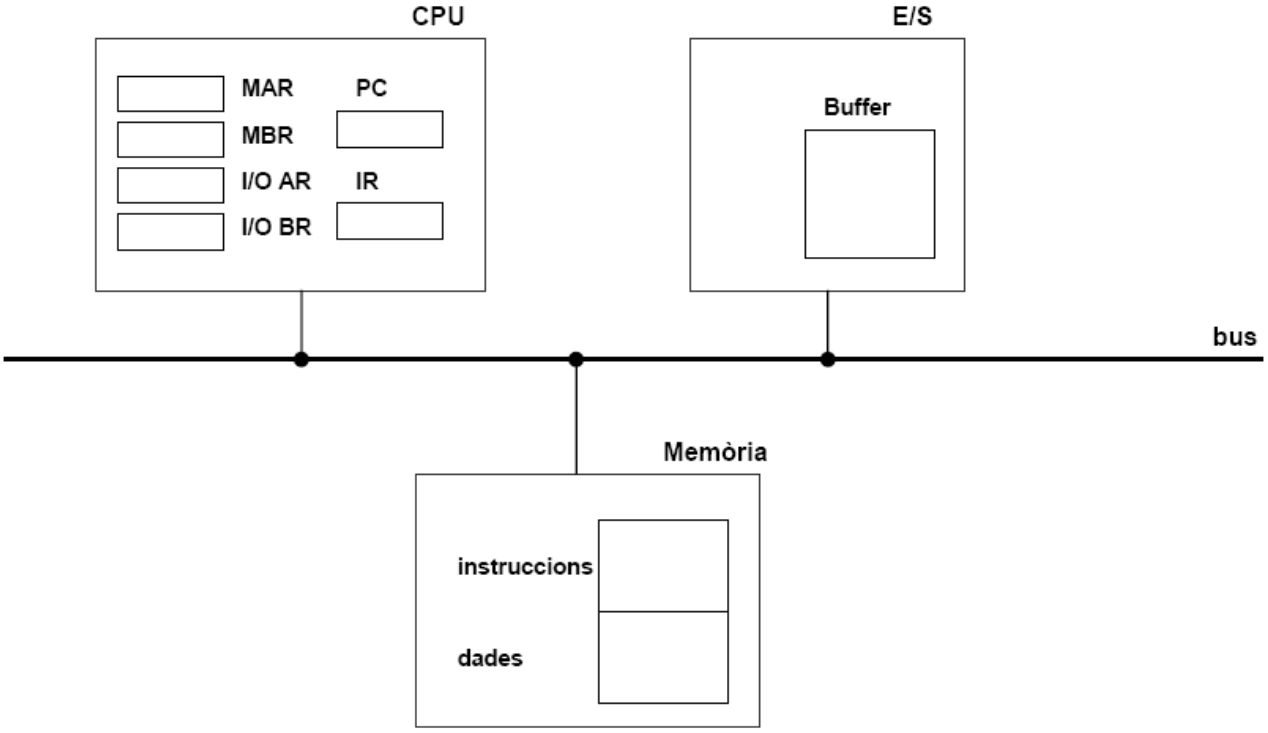
Tema 1. Introducció als Sistemes Operatius

Tema 1. Introducció als Sistemes Operatius

1. Introducció als ordinadors

1.1. Components bàsics

Una visió a alt nivell d'un computador seria la següent:



Per intercanvi memòria - CPU
- MAR: Registre d'Adreces de Memòria (Memory Adress Register)
- MBR: Registre de Buffer de Memòria (Memory Buffer Register)

Per intercanvi E/S - CPU
- I/O AR: Registre d'adreces d'E/S
- I/O BR: Registre de Buffer d'E/S

Per execució de programes
- PC: comptador de programa (Program Counter).
- IR: Registre d'Instrucció. Conté la instrucció carregada a la CPU

1. Processador: controla el funcionament del computador. Una de les seves funcions importants és l'intercanvi de dades amb memòria.
2. Memòria principal: emmagatzema dades i programes. Consta d'un conjunt de posicions definides mitjançant direccions o adreces numerades seqüencialment.
3. Mòduls d'Entrada/Sortida: transfereixen dades entre el computador i l'entorn extern. L'entorn extern està format per diversos dispositius, incloent-hi dispositius de memòria secundària (com els discos), equips de comunicacions i terminals. Els buffers són zones d'emmagatzemament internes al dispositiu que mantenen les dades fins que aquestes es puguin transferir.
4. Bus del sistema: canal de comunicació entre els diversos mòduls.

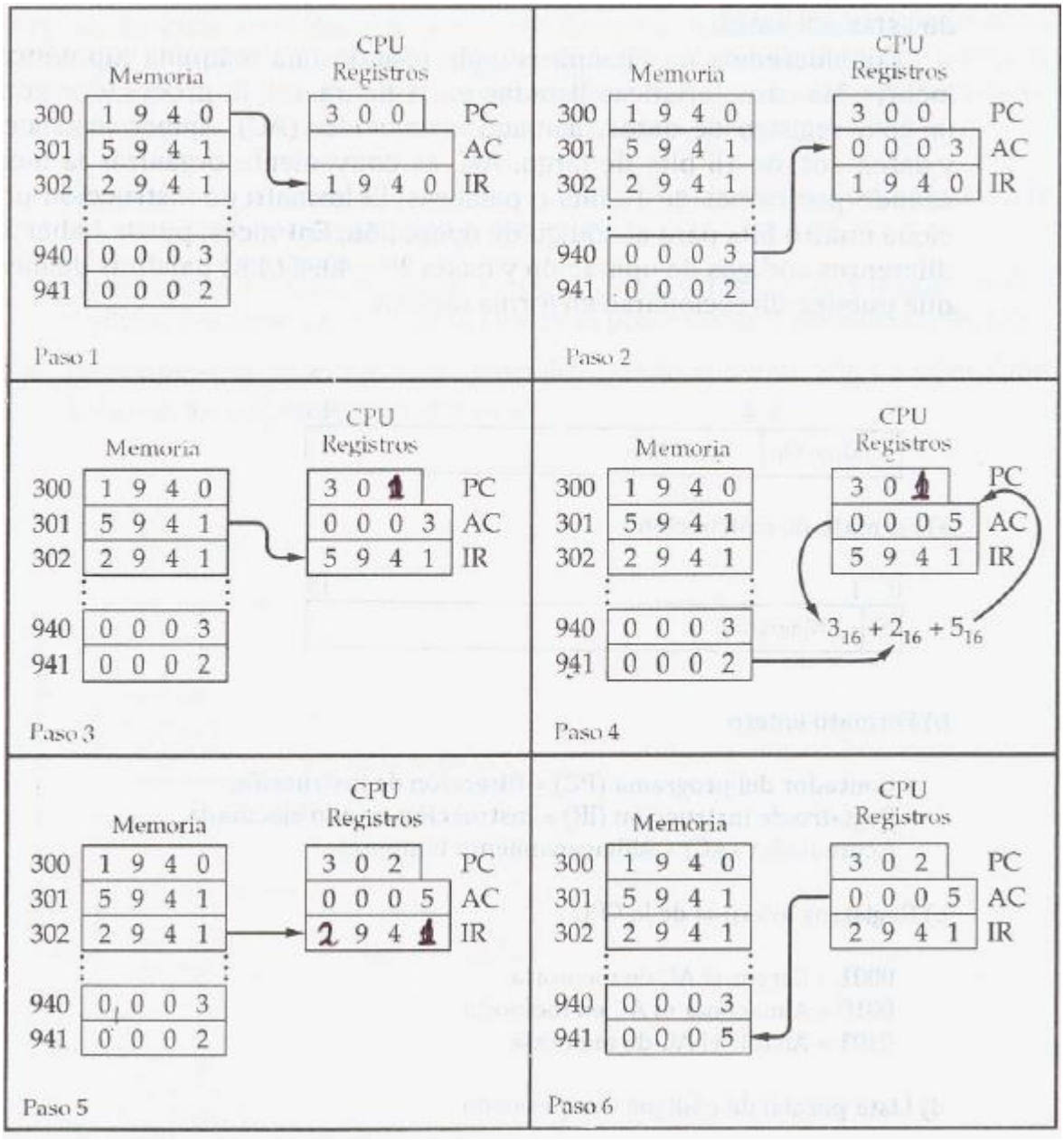
Existeixen 4 categories per classificar totes les accions que pot executar un programa:

a) Processador - memòria: transferència d'informació entre la CPU i la memòria o a l'inrevés.
b) Processador - E/S: transferència d'informació entre la CPU i un dispositiu d'E/S o a l'inrevés.
c) Processament de dades: operacions aritmètiques o lògiques sobre les dades.
d) Control: instruccions que impliquin un salt en la seqüència d'execució.

1.2. Execució d'instruccions

- El processador té un sol registre de dades: l'acumulador (AC)
- El processador té IR i PC clàssics.
- Instruccions i dades tenen una mesura de 16 bits.
- El format per les instruccions és de 4 primers bits per codi operació i 12 restants per adreçament.
- Codificació de les operacions:

0001 Carregar l'AC de memòria
0010 Emmagatzemar contingut d'AC a memòria
0101 Sumar l'AC amb memòria



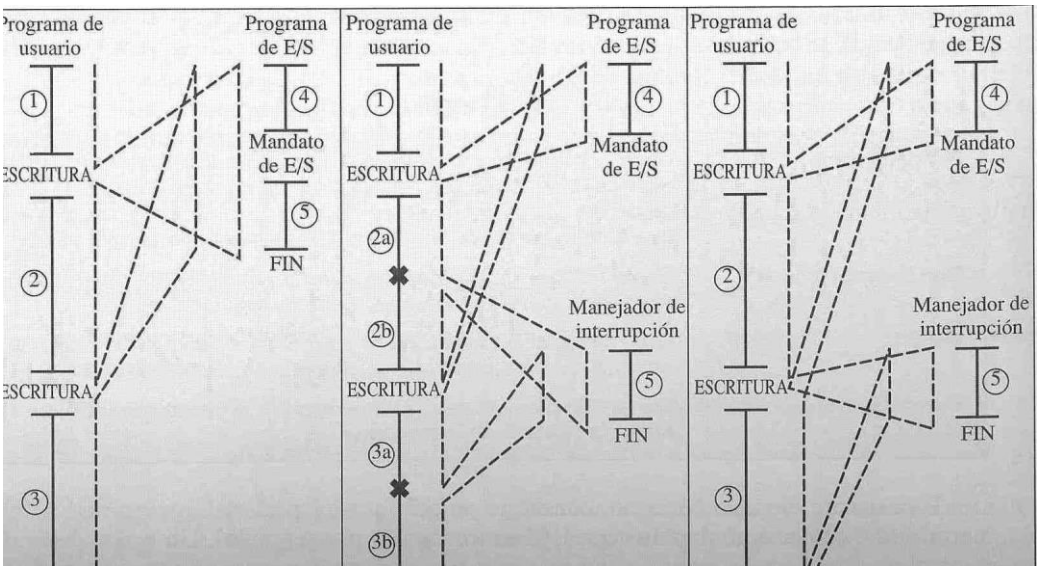
1.3. Interrupcions

Fins ara només hem mencionat i exemplificat transmissió de dades entre la memòria i la CPU. Però habitualment la transmissió s'efectua entre la CPU i un dispositiu d'E/S donat que la memòria és volàtil. També, pot efectuar-se una transmissió entre la memòria i un dispositiu d'E/S (DMA, com ja explicarem més endavant).

Hi ha dos maneres de comunicar-se amb el dispositiu des del sistema operatiu:

1. Polling: el programa esta constantment consultant el perifèric; per si ha finalitzat una transferència d'E/S. Per exemple: el teclat. La CPU hauria d'examinar a intervals regulars el dispositiu del teclat per saber si s'ha premut alguna tecla.

2. Interrupcions: la CPU segueix el seu treball normal i és el dispositiu que interromp a la CPU quan succeeix un event (al prémer una tecla del teclat).

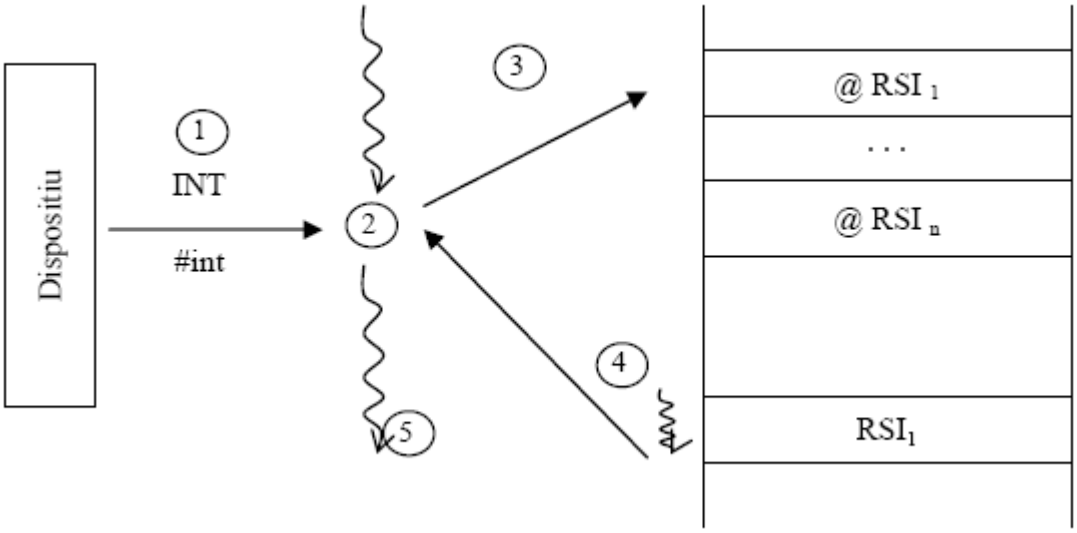


En el cas de les interrupcions, la CPU treballa com normalment i és el dispositiu d'E/S el que interrumpeix a la CPU quan hi ha algun succés (per exemple: al prémer una tecla).

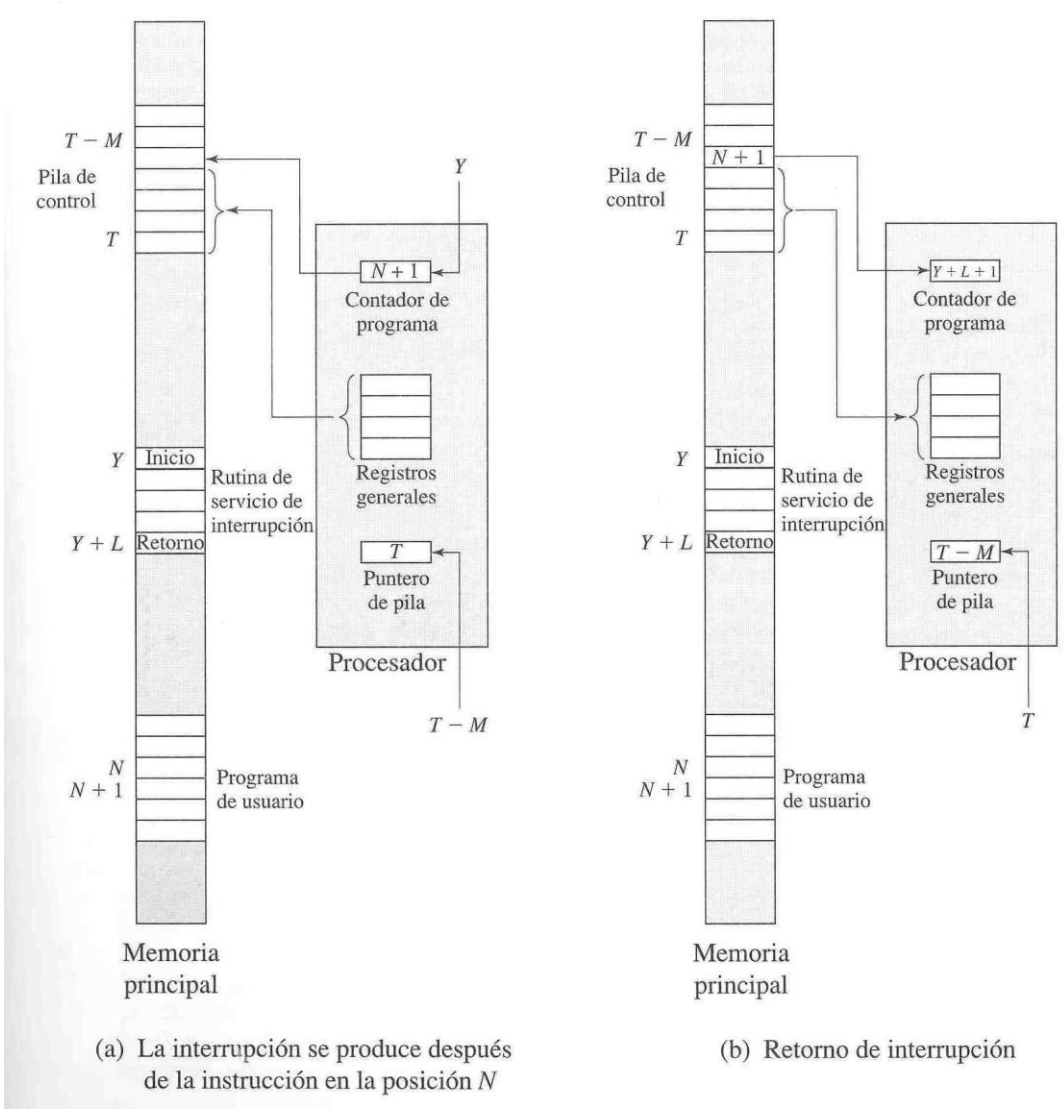
Quan això passa, la CPU transfereix el control a una posició fixa en memòria on espera trobar una rutina que sàpiga tractar la interrupció. Una variant més flexible és que la CPU utilitzi el número del dispositiu que causa la interrupció (IRQ) per indexar una taula que es troba en una posició determinada de memòria. En aquesta taula es troba l'adreça de la rutina que ha de tractar la interrupció. A aquesta rutina se la coneix amb el nom de Rutina de Servei d'Interrupció (RSI). A continuació realitza la tasca de la manera més breu possible, llavors retorna el control al programa que s'estava executant.

El cicle d'execució d'una interrupció és:

1. S'està executant un flux d'instruccions quan un dispositiu genera una interrupció. #int identifica el dispositiu.
2. Hem de guardar el context, es guarden un seguit de registres.
3. Mitjançant el #int, s'indexa la taula de vectors d'interrupcions l'@RSI de la rutina que s'haurà d'executar; programada prèviament d'acord amb el dispositiu.
4. Saltem a l'adreça de la RSI que hem obtingut de la taula. S'executa la RSI associada.
5. Retornem el control al flux inicial, recuperant el context guardat anteriorment



A efectes de memòria podem veure aquest processament de la interrupció i el canvi de context que es presenta en el següent esquema associat:



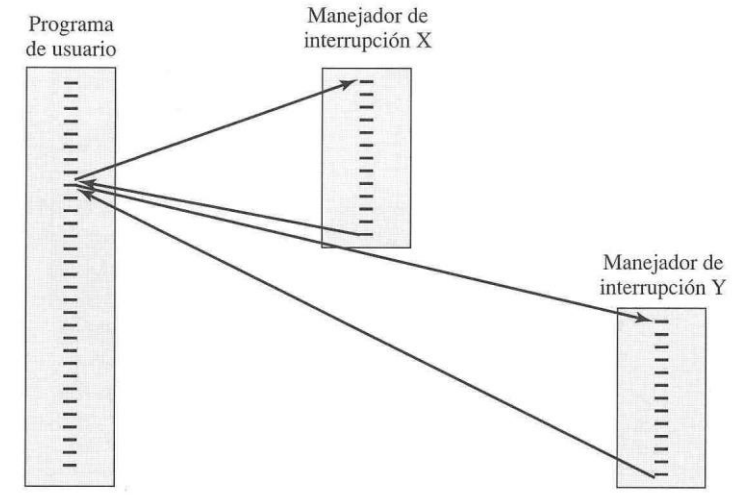
1.4. Múltiples interrupcions

Fins ara hem mostrat el processament d'interrupcions tenint en compte que apareix una única interrupció mentre executem un únic programa.

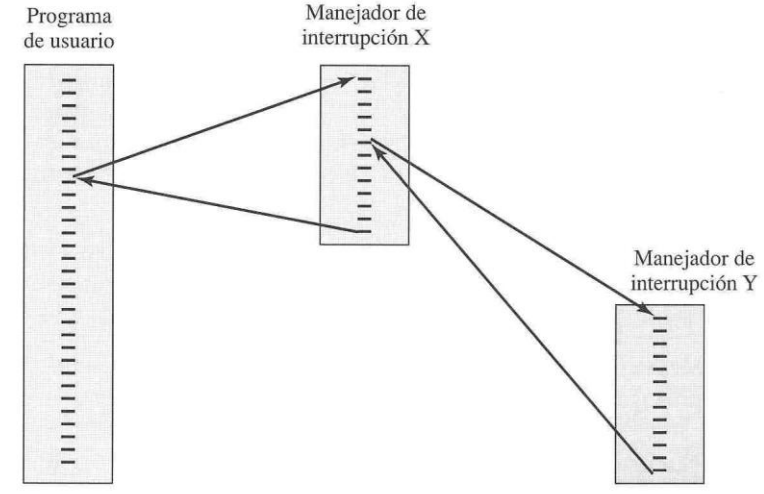
Però és freqüent que puguin aparèixer diverses interrupcions a la vegada. Per exemple, podem tenir un procés que mentre està imprimint estigui esperant dades per una línia de comunicacions. Així doncs ens pot arribar una interrupció de la línia de comunicacions quan estem tractant una interrupció de la impressora.

Es poden considerar dues alternatives per tractar aquesta situació:

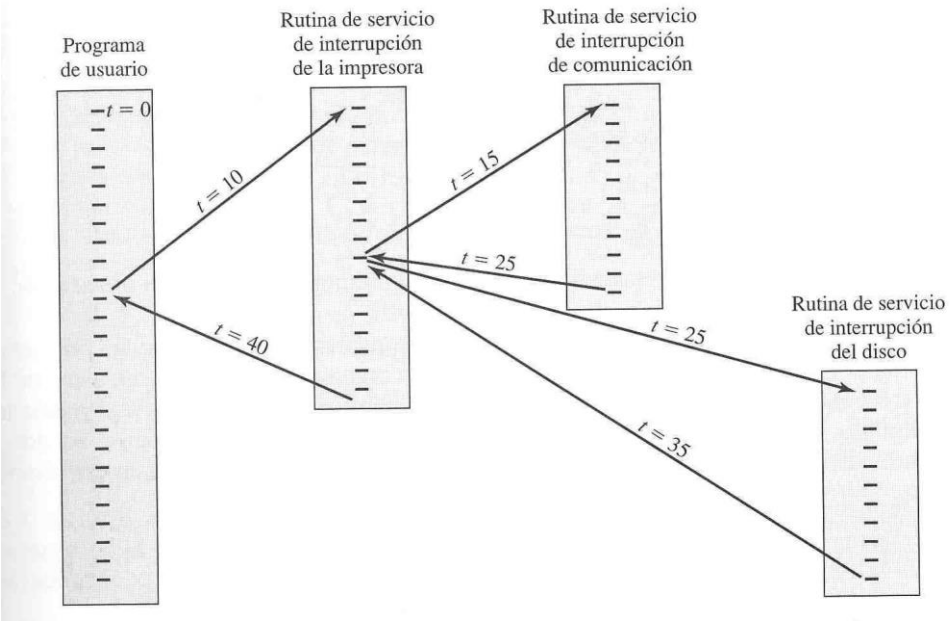
1. Inhabilitar les interrupcions mentre s'està processant una interrupció. Els avantatges són clars: és una estratègia vàlida, senzilla i que manté l'estricte ordre d'arribada per a processar les interrupcions. El principal inconvenient és que no podem prioritzar certes interrupcions.
2. Definir prioritats entre les interrupcions i permetre interrompre la RSI si la interrupció té una prioritat més alta.



(a) Procesamiento secuencial de interrupciones



(b) Procesamiento anidado de interrupciones



1.5. Multiprogramació

Però tot i les interrupcions, la utilització del processador pot ser completament ineficient. Si els temps d'espera de les E/S d'un programa són més altes que el temps de processament del mateix, gran part del temps el processador estarà sense fer res.

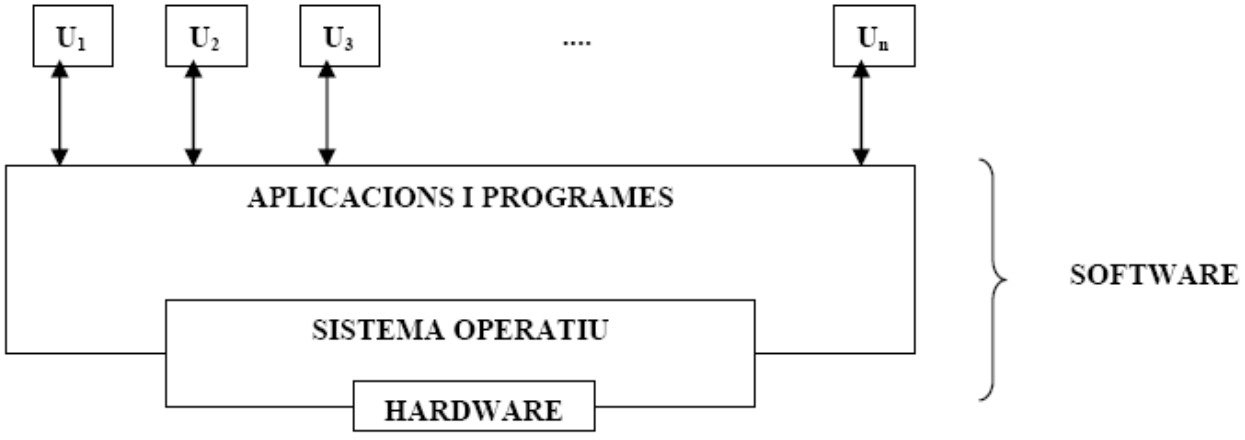
Per això és convenient poder executar diversos programes a la vegada per aprofitar millor el temps de CPU.

2. Ubicació del Sistema Operatiu en el sistema informàtic

Un sistema operatiu és un programa que controla l'execució de programes i que actua com a interfície entre els programes i el hardware de l'ordinador.

Se li solen demanar 3 característiques:

1. Facilitat d'ús
2. Eficiència
3. Capacitat per evolucionar



3. Funcions dels sistemes operatius

Podem agrupar les diverses funcionalitats que li demanem a un sistema operatiu en les següents 7:

1. Desenvolupament de programes: el SO ha de proporcionar eines (tals com editors, compiladors, depuradors, etc.) que facilitin la creació de programes.
2. Execució de programes: un sistema operatiu ha de permetre d'executar programes i que la seva execució sigui transparent a l'usuari (càrrega, dispositius E/S etc.). També ha de permetre la multiprogramació, ja sigui de manera concurrent o paral·lela.
3. Accés a dispositius E/S: el SO ha de donar una visió unificada per accedir als diversos dispositius que controla.
4. Accés controlat a fitxers.
5. Accés al sistema: per a sistemes compartit o públics, el SO controla l'accés al sistema complet i als diversos dispositius d'E/S.
6. Detecció i resposta a errors: durant una execució es poden produir multitud d'errors. Errors de hardware (errors de memòria, de mal funcionament d'un dispositiu d'E/S), errors de software (divisions per zero, core dumpeds, etc.), etc. En cada cas el SO ha de proporcionar una resposta adequada a l'error que es produeix.
7. Comptabilitat: un bon SO ha de recollir dades de rendiment del seu funcionament: temps de resposta, temps d'accés, ús d recursos, etc.

4. Evolució històrica dels sistemes operatius

4.1. Sistemes per lots.

Els primers ordinadors eren màquines que acostumaven a usar com a dispositius d'entrada lectors de targetes perforades i unitats de cinta; i com a dispositius de sortida impressores, perforadores de targeta i unitats de cinta. L'usuari no interactuava sinó que donava una entrada i després del processament d'aquesta per part de la màquina rebia una sortida.

Així doncs, el sistema operatiu d'aquestes màquines era força simple: s'encarregava de transferir el control d'un programa al següent. El SO sempre estava resident a memòria.

Per accelerar aquest processament els operadors de la màquina agrupaven en lots els treballs de característiques similars.

En aquest entorn de treball sovint es desaprofitava temps de CPU donat que les velocitats dels dispositius d'E/S són sempre inferiors a les velocitats del processador (exemple CPU treballaven executant milers d'instruccions per segon mentre que les lectores de targetes processaven 20 targetes per segon).

L'aparició dels discos va permetre al SO poder tenir tots els processos a executar en una sola ubicació (cosa que no passava amb la gestió seqüencial de les targetes perforades) i poder accedir a ells quan ho desitgés. D'aquesta manera podia planificar la seva execució per aprofitar el temps de CPU i el defasatge amb els dispositius d'E/S.

Apareix la capacitat de multiprogramació, on el SO s'encarrega de carregar diversos processos a memòria i va alternant-los per poder aprofitar el temps de CPU i evitar les esperes per culpa dels delays dels dispositius d'E/S. Això no és senzill ja que implica planificació de processos, gestió de la memòria, planificació de la CPU i l'administració de les E/S compartides.

4.2. Sistemes de temps compartit (online).

El següent pas evolutiu a la multiprogramació són els sistemes de temps compartit. Aquí el sistema operatiu no tan sols ha de executar diversos processos alhora sinó que ha de donar resposta a tots ells (interactuar) en un temps acceptable (un segon màxim aproximadament).

4.3. Sistemes per PCs.

Apareixen els PCs a la dècada de 1970. Els primers no tenien CPUs amb mecanismes per poder protegir els sistema operatiu dels programes d'usuari. Així doncs els SO eren monousuari i monotasca. Posteriorment han anat evolucionant cap entorns amigables coms les que coneixem (MS-DOS, el OS/2 d'IBM, diverses versions del Microsoft Windows, les diverses distribucions de Linux, etc.).

Els SO usats per PCs s'han aprofitat dels desenvolupaments de SO creats per mainframes. Veure Figura 1.4 pàgina 11 Silberschatz.

4.4. Sistemes paral·lels.

Actualment la majoria de ordinadors tenen un únic processador però hi ha una tendència, a tenir en compte, cap els sistemes multiprocessador. Aquests sistemes (coneguts també amb el nom de sistemes fortament acoblats) comparteixen bus, rellotge i, a vegades, memòria i perifèrics. Avantatges d'aquests sistemes:

- L'augment del rendiment (realitzar més treball en menys temps). Tot i que cal recordar que n processadors no fan accelerar n cops les tasques, ja que ens cal una tasca nova que s'encarregui de gestionar la cooperació entre processadors, i això requereix un temps addicional (overhead).
- Estalvi de costos ja que es comparteixen perifèrics (discos, sistemes de còpia de dades, etc.).
- Són més fiables, ja que si un processador cau les tasques que executava aquest poden ser traslladades a un dels altres processadors sense haver d'aturar el sistema, només anirà més lent (degradació suau).

Els sistemes de processadors múltiples poder usar dos tipus de processament:

- Multiprocessament simètric (symmetric multiprocessing, SMP): el més comú. Cada CPU executa una còpia idèntica del sistema operatiu de manera concurrent i aquestes es comuniquen en cas de necessitat. Avantatge: execució d'N processos a la vegada. Cal cuidar les operacions d'E/S de cada procés. Tots els SO moderns (Windows NT, Solaris, UNIX, Linux) ofereixen suport per a SMP.
- Multiprocessament asimètric: cada CPU se li assigna una tasca específica. Un processador mestre controla el sistema (relació mestre-esclau).

Cal no confondre paral·lelisme i concurrència:

- Concurrència: en un moment donat només hi ha una tasca executant-se ja que només tenim un processador.
- Paral·lelisme: en un moment poden haver-hi N tasques executant-se en N processadors diferents.

4.5. Sistemes de temps real.

S'utilitzen per aplicacions de control (industrials...). La principal característica d'aquests sistemes operatius és que asseguren un temps de resposta màxim pels successos (events) hardware.
Han de complir unes restriccions de temps molt estrictes. Tot el codi del Sistema Operatiu pot ocupar 40 K, és molt petit. Exemple: Sistema Operatiu gestió ABS en un cotxe.

4.6. Sistemes distribuïts.

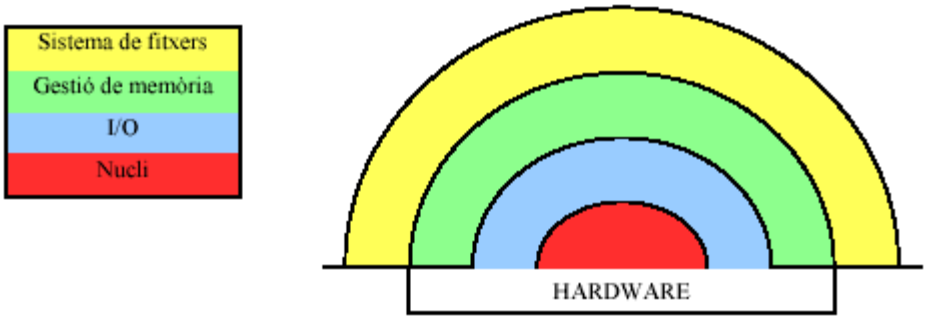
Sistema distribuït: conjunt de màquines independents connectades mitjançant una xarxa d'interconnexió que, gràcies al Sistema Operatiu es veuen com una sola màquina.

Avantatges: són més econòmics, tenen més potència de càlcul, són més fiables i també més escalables.

Inconvenients: poden provocar saturació de la xarxa, són menys segurs o més vulnerables a atacs i hi ha poc software desenvolupat.

5. Components d'un sistema Operatiu

Un sistema operatiu està format, bàsicament, de 4 mòduls o subsistemes:



- NUCLI: és l'única capa que pot inhibir interrupcions. Les seves funcions són:
  - Commutació de CPU entre les diferents tasques.
  - Gestió i control de les interrupcions.
  - Sincronització entre les diferents tasques.

- SISTEMA E/S: és la única capa que pot executar instruccions del tipus in/out. La seva funció és la comunicació amb els perifèrics. Per tant, les capes superiors per dialogar amb el hardware, ho han fer a través d'aquesta capa.

- GESTIÓ DE MEMÒRIA: conjuntament amb el hardware, crea la possibilitat de disposar de memòria virtual. A més a més, aquesta capa és la responsable de garantir (també conjuntament amb el hardware) la protecció de les dades a memòria.

- SISTEMA DE FITXERS: proveeix una visió estructurada de les dades emmagatzemades al disc. Fins i tot pot ser capaç de deixar veure certs perifèrics com a fitxers.

Les aplicacions d'usuari són programes creats per l'usuari. Quan aquests programes necessiten alguna funcionalitat del sistema operatiu, realitzen el que s'anomena una "crida al sistema", que consisteix en fer crides a funcions que ofereixen les diferents capes. És a dir, si l'aplicació vol mostrar un caràcter per pantalla, doncs haurà de cridar a una funció de la capa E/S per aconseguir-ho.