Sistema d'entrada/sortida

Miquel Albert Orenga
Gerard Enrique Manonellas

PID_00218251

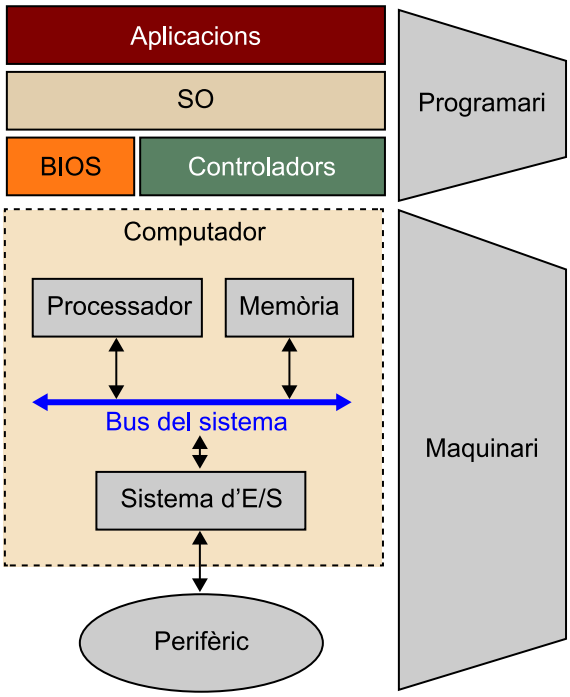
Universitat Oberta de Catalunya

Índex

Introducció

Tot computador necessita dur a terme intercanvi d'informació amb persones o altres computadors mitjançant uns dispositius que anomenem de manera genèrica **dispositius perifèrics**. Per a fer una operació d'E/S entre el computador i un perifèric, cal connectar aquests dispositius al computador i gestionar de manera efectiva la transferència de dades. Per a fer-ho, el computador disposa del **sistema d'entrada/sortida (E/S)**.

Aquest sistema d'E/S és la interfície que té el computador amb l'exterior i l'objectiu que té és facilitar les operacions d'E/S entre els **perifèrics** i la **memòria** o els **registres del processador**. Per a gestionar les operacions d'E/S és necessari un maquinari i el suport d'un programari.



Atesa la gran varietat de perifèrics és necessari dedicar un maquinari i un programari específics per a cadascun. Per aquest motiu s'ha intentat normalitzar la interconnexió dels perifèrics i el computador mitjançant el que s'anomena **mòduls d'E/S** o **controladors d'E/S**. Això ens permet tenir, per una banda, una connexió, entre el mòdul d'E/S i el perifèric, específica i amb unes característiques pròpies que difícilment es poden generalitzar per a utilitzar-les en altres dispositius, i per altra banda, una connexió entre els mòduls d'E/S i el computador comuna a tots els controladors, però aquests mòduls, a més a més de permetre la connexió dels perifèrics al computador, disposen de la lògica necessària per a tenir certa capacitat de processament i gestionar les transferències d'informació.

Cal tenir present que la gestió global del sistema d'E/S d'un computador la fa el sistema operatiu (SO). Les tècniques per a controlar aquest sistema d'E/S les utilitza el SO i el programador quan volen accedir al perifèric, però en les màquines actuals, a causa de la complexitat de controlar i gestionar els perifèrics, l'accés es fa generalment mitjançant crides al SO, que és qui gestiona la transferència. El conjunt de rutines que permeten controlar un determinar perifèric és el que anomenem habitualment **programes controladors** o ***drivers*** i quan el SO vol fer una operació d'E/S amb un perifèric crida a una d'aquestes rutines.

Aquest mòdul se centra en l'estudi del sistema d'E/S i parlarem de les principals tècniques utilitzades i quines característiques ha de tenir el maquinari i el programari necessari per a gestionar les diferents maneres de fer la transferència d'informació entre el computador i el perifèric.

Objectius

Amb l'estudi d'aquest mòdul es pretén que l'estudiant assoleixi els objectius següents:

1. Conèixer els aspectes bàsics del sistema d'E/S d'un computador.

2. Aprendre les tècniques bàsiques d'E/S.

3. Entendre els avantatges de cadascuna d'aquestes tècniques per a millorar el rendiment del computador.

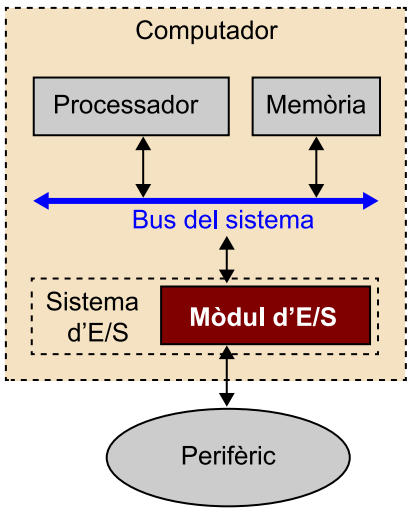
4. Tenir uns coneixements bàsics dels tipus de dispositius que podem connectar al computador i com es comuniquen amb el processador mitjançant el sistema d'E/S.

1. Aspectes bàsics de l'E/S

Quan parlem d'E/S d'informació entre un computador i un perifèric ho fem sempre des del punt de vista del computador. Així, diem que és una **transferència d'entrada** quan el perifèric és l'emissor de la informació i té com a receptor el computador (processador o memòria) i diem que és una **transferència de sortida** quan el computador és l'emissor de la informació i té com a receptor el perifèric.

De manera més concreta, tota operació d'E/S que es du a terme entre el computador i un perifèric és sol·licitada i governada des del *processador*, és a dir, és el processador qui determina en quin moment s'ha de fer i amb quin perifèric, si l'operació és de lectura o escriptura, quines dades s'han de transferir, i també qui dóna l'operació per acabada.

Per a dur a terme l'operació d'E/S hem de connectar el perifèric al computador. Per a fer-ho, cal que el computador disposi d'uns dispositius intermedis per on ha de passar tota la informació que intercanvia el computador amb el perifèric i que ens permet fer una gestió i un control correctes de la transferència. Aquests dispositius els anomenem de manera genèrica **mòdul d'E/S**.



Pot semblar lògic connectar el perifèric directament al bus del sistema del computador, però aquesta opció no és factible bàsicament per dues raons:

- La necessitat de gestionar una gran varietat de perifèrics amb unes característiques molt especifiques i diferenciades. Això fa molt complex afegir la lògica necessària dins del processador per a gestionar aquesta gran diversitat de dispositius.

- La diferència de velocitat entre si, en què, llevat de casos excepcionals, el processador és molt més ràpid que el perifèric. Per un costat, cal assegurar que no es perdin dades i, per l'altre, garantir principalment la màxima eficiència del processador, però també dels altres elements del computador.

Així, doncs, per a fer una operació d'E/S, el mòdul d'E/S ens ha de permetre establir, per una banda, **mecanismes de control** per a determinar l'inici i el final de l'operació d'E/S, la quantitat d'informació que cal transmetre, la detecció d'errors, etc., i per altra banda, **mecanismes per a fer la transferència de dades** considerant aspectes com la manera d'adreçar el perifèric, la conversió sèrie/paral·lel de la informació, la conversió de codis, la sincronització, etc. Aquests mecanismes es reparteixen entre la unitat de control del processador, el mòdul d'E/S i els programes d'E/S.

Quan volem fer l'operació d'E/S, ens cal diferenciar el cas d'una transferència individual, en què es transmet una sola dada i el control de la transferència és molt simple (llegir una tecla, mirar si s'ha fet un clic al ratolí), i la transferència de blocs, que es basa en un seguit de transferències individuals i cal un control més gran de tot el procés (llegir un fitxer, actualitzar el contingut de la pantalla).

Un altre aspecte important que cal considerar, atès que podem tenir connectats al computador una gran varietat de perifèrics, és que si es desencadenen operacions d'E/S de manera simultània el sistema d'E/S del computador ha de disposar dels mecanismes necessaris per a gestionar-les sense que es perdin dades.

1.1. Estructura del sistema d'E/S del computador

Els elements principals que formen el sistema d'E/S són els següents:

- els perifèrics,
- els mòduls d'E/S,
- els sistemes d'interconnexió externs i
- el mapa de memòria i instruccions d'E/S.

A continuació farem una descripció breu d'aquests elements i com interactuen entre si.

1.1.1. Perifèrics

Els perifèrics són dispositius que es connecten al computador mitjançant els mòduls d'E/S i que serveixen per a emmagatzemar informació o per a dur a terme un tipus determinat de comunicació amb l'exterior amb humans, amb màquines o amb altres computadors.

La classificació més habitual és la següent:

- Per a la interacció amb humans:
  - Entrada.
  - Sortida.

- Per a la interacció amb altres computadors o sistemes físics (en què les operacions que es fan són generalment d'E/S):
  - Emmagatzematge.
  - Comunicació.

En un perifèric distingim habitualment dues parts: una part mecànica i una part electrònica. La part mecànica fa funcionar els elements principals que formen el perifèric, com és ara el motor per a fer girar un disc o moure el capçal d'una impressora, el botó d'un ratolí o el làser d'un dispositiu òptic. La part electrònica ens permet, per una banda, generar els senyals elèctrics per a gestionar els elements mecànics i, per altra banda, fer la conversió de les dades provinents del computador a senyals elèctrics o a l'inrevés.

La connexió física entre un perifèric i el computador es fa mitjançant el que anomenem **sistema d'interconnexió d'E/S**. Aquest sistema d'interconnexió d'E/S ens permet fer la gestió dels senyals de control, d'estat i de dades necessaris per a dur a terme una transferència d'informació que, com veurem més endavant, és gestionada des del mòdul d'E/S del computador.

En aquest mòdul ens centrarem a analitzar la transferència d'informació entre un perifèric i el computador mitjançant els mòduls d'E/S.

Nota

En aquest mòdul no analitzarem l'estructura i el funcionament del perifèric, i n'hi haurà prou de veure el perifèric com un element capaç d'enviar o rebre una determinada quantitat d'informació, a una determinada velocitat, mitjançant el sistema d'interconnexió d'E/S. No analitzarem tampoc les característiques tècniques que permeten al perifèric tenir aquestes prestacions.

La quantitat d'informació que pot enviar o rebre el perifèric per unitat de temps l'anomenem **velocitat de transferència** i generalment s'expressa en bits o bytes per segon.

La velocitat de transferència pot anar d'uns pocs bits per segon a *gigabytes* per segon, però hem de tenir present que un computador pot arribar a treballar a velocitats força superiors i hem de garantir que durant una transferència no es perdin dades.

1.1.2. Mòduls d'E/S

Un mòdul d'E/S és un controlador d'un o diversos perifèrics que estableix una interfície entre el perifèric i el computador (processador i memòria) per a facilitar la comunicació entre l'un i l'altre de manera que bona part dels detalls tècnics del perifèric quedin ocults a la resta del computador.

Del mòdul d'E/S distingim tres parts bàsiques:

1) Una interfície interna normalitzada amb la resta del computador mitjançant el bus de sistema que ens dóna accés al banc de registres del mòdul d'E/S.

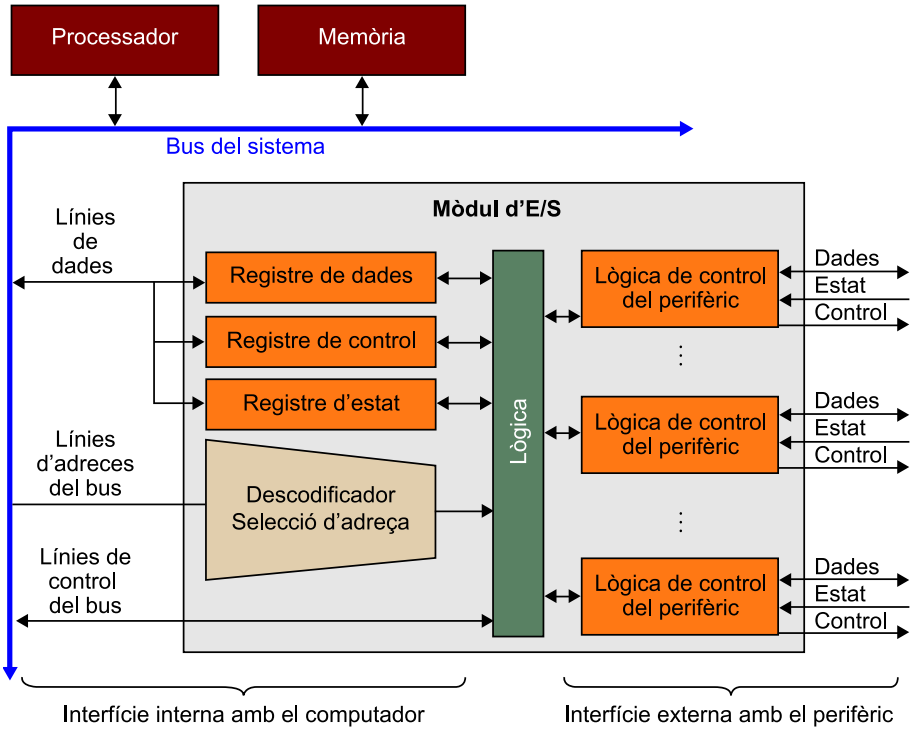
2) Una interfície externa específica per al perifèric que controla. Habitualment la connexió amb el perifèric es fa mitjançant un sistema d'interconnexió normalitzat d'E/S.

3) La lògica necessària per a gestionar el mòdul d'E/S. És responsable del pas d'informació entre la interfície interna i externa.

Nota

La complexitat d'un mòdul d'E/S pot variar molt. Per aquest motiu aquí farem una descripció general de les parts i característiques bàsiques més importants. En veurem altres característiques més específiques quan analitzem les diferents tècniques d'E/S.

A la següent figura podeu veure l'esquema general d'un mòdul d'E/S.

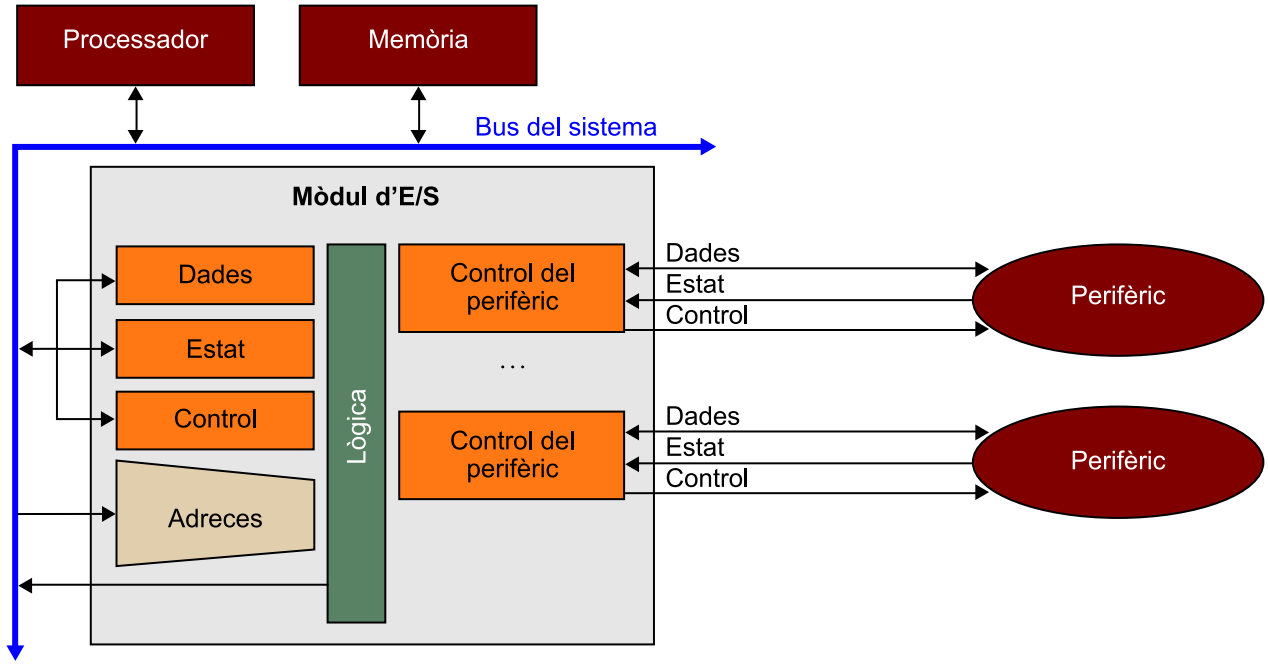


La forma de comunicació entre el mòdul d'E/S i el perifèric és especifica per a cada perifèric. Lògicament depèn de les característiques del perifèric que volem controlar, però també del sistema d'interconnexió utilitzat per a comunicar-se. Aquesta connexió té habitualment unes especificacions normalitza-

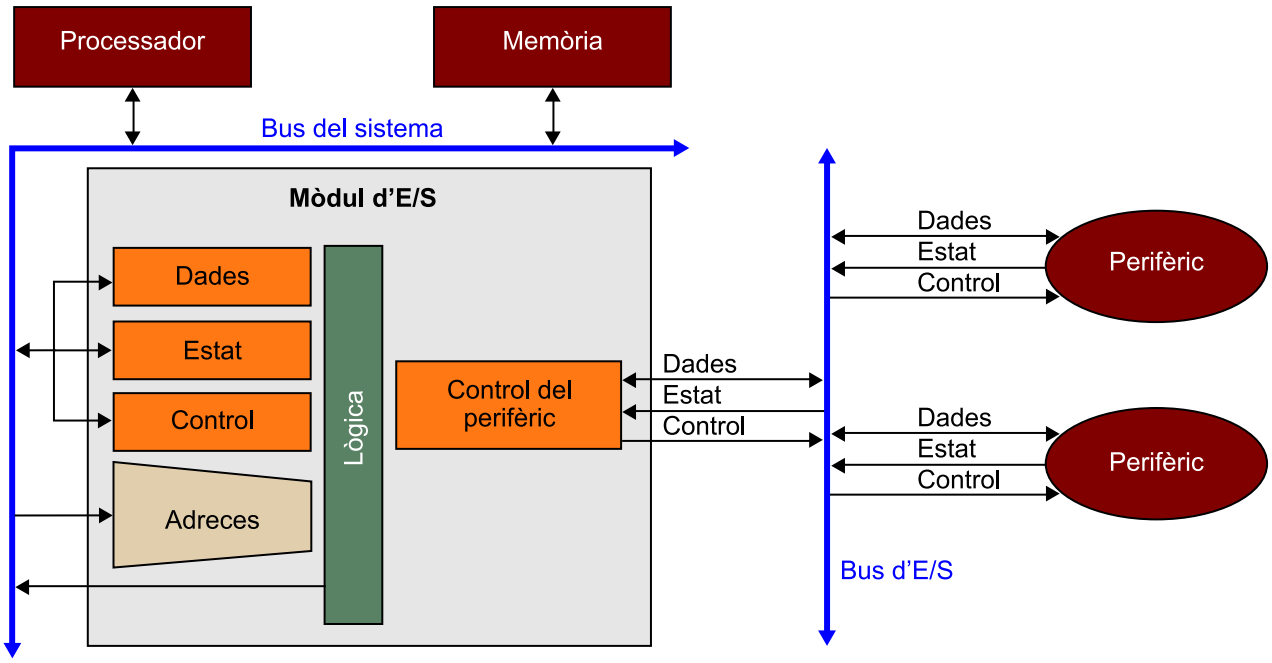
des i adaptades al tipus de transferència que s'ha de fer i l'anomenem **sistema d'interconnexió d'E/S**. Això fa que la interfície externa tingui unes característiques pròpies que difícilment es poden generalitzar.

Quan un mòdul d'E/S gestiona més d'un perifèric, hi ha dues configuracions bàsiques, la connexió punt a punt i la multipunt, tot i que les configuracions que trobem en màquines reals són molt variades. En la connexió punt a punt el mòdul d'E/S gestiona la comunicació amb cada perifèric individualment; no és un bus d'E/S però sí que té unes especificacions normalitzades de la connexió de manera semblant a les d'un bus normalitzat d'E/S. En la connexió multipunt el mòdul d'E/S gestiona la comunicació amb els perifèrics mitjançant un bus normalitzat d'E/S i hi cal afegir la lògica per a accedir al bus.

Connexió punt a punt entre el mòdul d'E/S i el perifèric



Connexió multipunt entre el mòdul d'E/S i el perifèric



La comunicació entre els mòduls d'E/S i el computador és sempre la mateixa per a tots els mòduls. Aquesta comunicació s'estableix mitjançant el bus del sistema de manera que el processador veu el mòdul d'E/S com un espai de memòria, però aquestes adreces, físicament, corresponen (estan mapades) a cadascun dels registres que té el mòdul d'E/S del computador i s'anomenen habitualment **ports d'E/S**. D'aquesta manera aconseguim que la comunicació entre el computador i el mòdul d'E/S es dugui a terme mitjançant instruccions de transferència per a llegir i escriure en els seus registres, d'una manera molt semblant a la manera com fem per accedir a la memòria.

Aquests registres es poden agrupar segons el tipus de senyals o el tipus d'informació que necessitem per a fer una gestió correcta del perifèric:

- Registres de control.
- Registres d'estat.
- Registres de dades.

Per a gestionar la comunicació entre el processador i el mòdul d'E/S són necessaris diferents tipus de senyals.

Els **senyals de control** els utilitzem generalment per a donar ordres al mòdul d'E/S, com és ara començar o parar una transferència, seleccionar modes d'operació del perifèric o indicar accions concretes que ha de fer el perifèric com comprovar si està disponible. Aquests senyals es poden rebre directament de les línies de control del bus del sistema o de les línies de dades del bus del sistema i s'emmagatzemen en el *registre de control*.

Els **senyals d'estat** ens donen informació de l'estat del mòdul d'E/S, com és ara saber si el mòdul està disponible o està ocupat, si hi ha una dada preparada, si s'ha acabat una operació, si el perifèric està engegat o parat, quina operació està fent, o si s'ha produït algun error i quin tipus d'error. Aquests senyals s'actualitzen generalment mitjançant la lògica del mòdul d'E/S i s'emmagatzemen en el *registre d'estat*.

Les **dades** són la informació que volem intercanviar entre el mòdul d'E/S i el processador mitjançant les línies de dades del bus del sistema i s'emmagatzemen en el *registre de dades*.

Les **adreces** les posa el processador al bus d'adreces i el mòdul d'E/S ha de ser capaç de reconèixer aquestes adreces (adreces dels ports d'E/S) corresponents als registres d'aquest mòdul. Per a saber si l'adreça correspon a un dels registres del mòdul utilitzem un descodificador. Aquest descodificador pot formar part del mòdul d'E/S o de la mateixa lògica del bus del sistema.

Cal tenir present que un computador pot tenir definits diferents tipus de connexions normalitzades entre el mòdul d'E/S i la resta del computador. Tant el mòdul d'E/S com el computador s'han d'adaptar a aquests tipus de connexió de manera que tenim mòduls d'E/S adaptats a les diferents normes, i això té implicacions pel que fa al maquinari i a la manera de gestionar les operacions d'E/S com veurem més endavant quan analitzem les tècniques bàsiques d'E/S.

1.1.3. Sistemes d'interconnexió externs

En un computador distingim dos tipus bàsics de sistemes d'interconnexió: els interns del computador, que ens permeten connectar el processador, la memòria i el sistema d'E/S i que anomenem **bus del sistema**, i els externs al computador, que ens permeten connectar el sistema d'E/S amb els diferents perifèrics i que anomenem **sistemes d'interconnexió d'E/S** o **busos d'E/S**.

Des del punt de vista del sistema d'E/S, el **bus del sistema** ens permet la comunicació entre els mòduls d'E/S i la resta del computador. Aquest bus té un estructura jeràrquica formada per diferents tipus de busos per a aïllar els elements més ràpids del més lents i d'aquesta manera millorar les prestacions del sistema.

Els **sistemes d'interconnexió d'E/S** o **busos d'E/S** ens permeten la comunicació dels mòduls d'E/S amb els perifèrics o dispositius amb prou autonomia per a gestionar una operació d'E/S i els mòduls d'E/S. Les característiques d'aquests sistemes s'adapten al tipus de dispositius que hi hem de connectar.

Físicament un sistema d'interconnexió està format per un conjunt de fils conductors o línies que interconnecten diferents dispositius. Per aquestes línies hi circulen senyals elèctrics que els dispositius que hi tenim connectats poden interpretar com a senyals binaris. Hi ha tres tipus de senyals bàsics: senyals de dades, d'adreces i de control.

Les següents són les característiques principals dels sistemes d'interconnexió externs:

- **Amplada de banda**: la quantitat màxima d'informació que podem transmetre per unitat de temps. S'expressa en bits o bytes per segon.

- **Sèrie/paral·lel**: en una interconnexió paral·lela hi ha diverses línies que connecten el mòdul d'E/S i el perifèric i hi poden transmetre diversos bits simultàniament mitjançant les línies de dades. En una interconnexió sèrie només hi ha una línia per a transmetre les dades i els bits s'han de transmetre un a un. Tradicionalment les interconnexions de tipus sèrie eren per a dispositius lents i les de tipus paral·lel per a dispositius més ràpids, però amb les noves generacions de sistemes d'interconnexió sèrie d'alta velocitat els paral·lels cada cop són menys utilitzats.

- **Punt a punt / multipunt**: una interconnexió punt a punt té un enllaç dedicat entre el mòdul d'E/S i el perifèric. En una interconnexió multipunt, que habitualment s'anomena *bus d'E/S* i que disposa d'un enllaç compartit entre diferents perifèrics i el mòdul d'E/S, el fet de tenir múltiples dispositius connectats a un mateix conjunt de línies fa necessari establir mecanismes per a controlar-ne l'accés.

Altres característiques típiques dels busos d'E/S són:

- **Mode d'operació síncron/asíncron/semisíncron**: si el control dels accessos al bus és controlat o no per un rellotge.

- **Multiplexació de dades i adreces**: si les línies del bus estan dedicades a dades i adreces o si es comparteix les mateixes línies per a dades i per a adreces.

- **Arbitratge centralitzat i distribuït**: és centralitzat quan un únic àrbitre o controlador determina qui ha d'accedir al bus a cada moment i és distribuït quan els dispositius connectats al bus disposen de capacitat de controlar l'accés al bus.

- **Tipus d'operacions de lectura/escriptura**: diferents maneres de fer les operacions de lectura i escriptura, com és ara la transferència de blocs o la combinació d'operacions de lectura i escriptura.

Normalització d'un bus

La normalització d'un bus consisteix a donar una descripció detallada i precisa de les característiques que té a diferents nivells. Els nivells principals són el mecànic (mides i connectors), l'elèctric (tipus de senyals elèctrics que circulen per les línies), el lògic (descripció de tots els senyals: adreces, dades i control) i el cronograma (quina és la seqüència de senyals per a fer la transferència d'una dada o d'un bloc).

La normalització d'un bus en facilita la divulgació i també el disseny dels dispositius que hi hem de connectar. Generalment la majoria de normalitzacions són definides per organismes internacionals. El més conegut en l'àmbit de l'electricitat i l'electrònica és l'IEEE.

- **Esquema d'adreçament**: n'hi ha de dos tipus bàsics: l'*adreçament lògic*, que és quan l'espai d'adreçament de memòria és comú a tots els dispositius i cadascun disposa d'un rang d'adreces únic i els dispositius per a descodificar l'adreça per a saber si aquesta adreça és dins del seu rang; l'*adreçament geogràfic*, que és quan cada dispositiu té una adreça pròpia i se separa la identificació del mòdul de la selecció de l'adreça dins del mòdul.

1.1.4. Mapa de memòria i instruccions d'E/S

Tal com hem explicat, el processador veu el banc de registres del mòdul d'E/S com un espai de memòria adreçable, de manera que cada registre del mòdul d'E/S té associada (mapada) una adreça única. Vegem ara com hem d'accedir a aquestes adreces, que anomenem **ports d'E/S**.

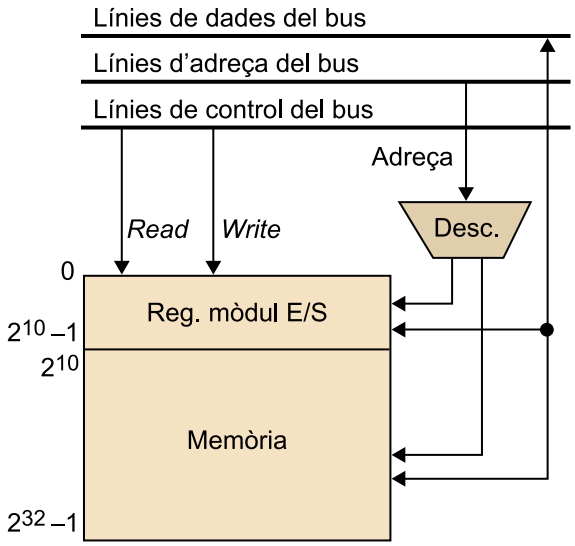
Per a identificar els registres del mòdul d'E/S hi ha dues possibilitats:

1) Mapa comú de memòria i E/S. No hi ha distinció entre adreces de memòria i registres d'E/S. Per a accedir als registres s'utilitza descodificadors que s'activen a partir de les línies del bus d'adreces i es fa servir els mateixos senyals de control (READ/WRITE) que s'utilitza per a seleccionar la memòria. Podem accedir als ports d'E/S amb les mateixes instruccions de transferència que utilitzem per a accedir a memòria (MOV i les variants que té).

Aquest sistema té l'avantatge que ens permet aprofitar l'ampli conjunt d'instruccions de què disposa el processador per a accedir a memòria i podem fer programes més eficients. El principal desavantatge és que hem de dedicar una part del valuós espai de memòria a les adreces d'E/S i cal ser curosos amb la quantitat d'espai assignat ja que aquest espai va en detriment de l'espai de memòria disponible, però cada cop aquest problema és menys important a causa de l'increment de l'espai de memòria adreçable.

Exemple de mapa comú de memoria

A la següent figura teniu un exemple de connexió d'una memòria de 2^{32} adreces i 2^{10} ports d'E/S utilitzant un mapa comú i un mapa independent de memòria i E/S.

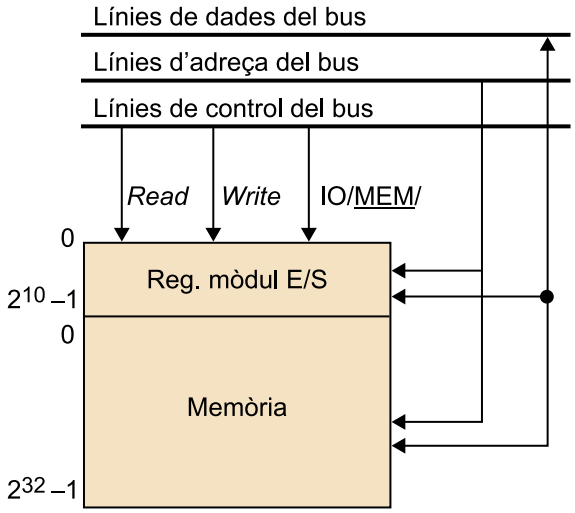


2) Mapa independent d'E/S. Hi ha distinció entre adreces de memòria i registres d'E/S. Les línies d'adreces s'acostumen a compartir, però hi cal afegir algunes línies de control per a distingir si un accés és a memòria o a un port d'E/S. També són necessàries instruccions especifiques d'E/S. Les instruccions utilitzades habitualment són IN (per a llegir del port d'E/S) i OUT (per a escriure en el port d'E/S).

Aquest sistema té l'avantatge que la memòria disposa de tot el rang d'adreces i el clar desavantatge que disposa d'un reduït nombre d'instruccions especifiques d'E/S que només disposen dels modes d'adreçament més bàsics per a accedir als ports d'E/S.

Exemple de mapa independent

A la següent figura teniu un exemple de connexió d'una memòria de 2^{32} adreces i 2^{10} ports d'E/S utilitzant un mapa independent de memòria i E/S.



Una qüestió interessant que cal considerar és que si fem servir instruccions específiques d'E/S el processador pot saber en la fase de descodificació de la instrucció que farem una operació d'E/S. Això fa més fàcil la gestió dels senyals

de control per a dur a terme aquesta operació i també fa més fàcil establir un mecanisme de protecció per a controlar que aquestes instruccions només es puguin executar en mode supervisor. Si no tenim instruccions especifiques d'E/S, fins que no es fa l'accés al port d'E/S (es descodifica l'adreça) no podem saber si aquell accés s'ha de permetre o no i això en complica la gestió.

1.2. Operació d'E/S

Per a garantir que una transferència de dades es fa amb èxit cal definir una sèrie de passos i establir una sèrie de mecanismes que ens permetin controlar en tot moment l'operació d'E/S.

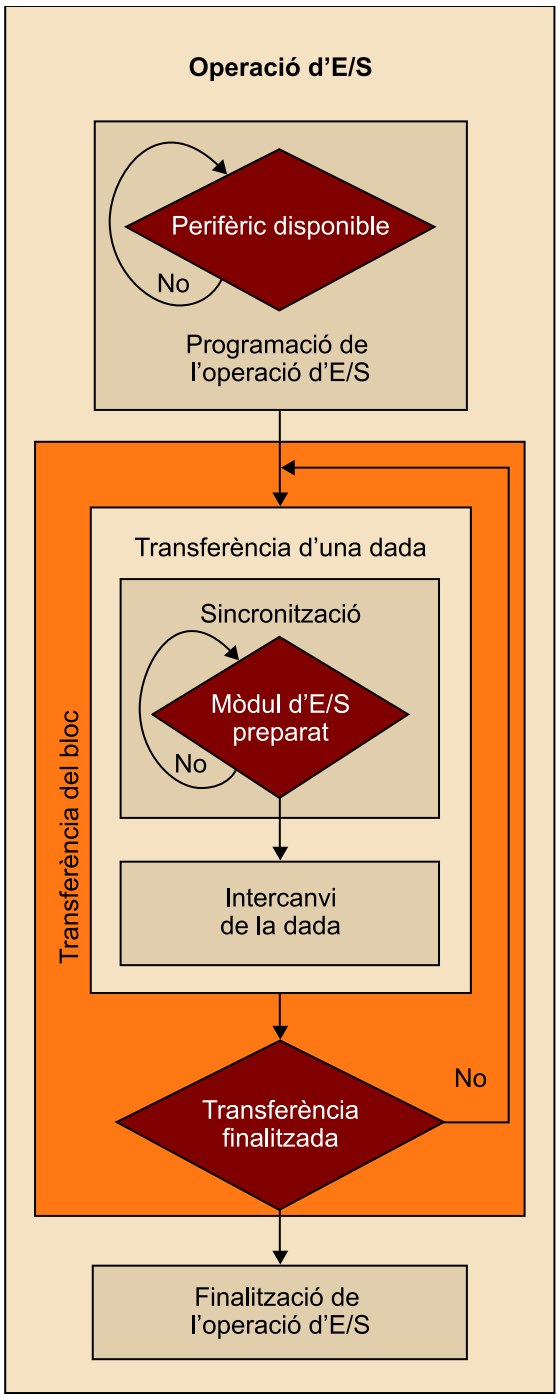
En el procés global d'una operació d'E/S distingim els passos següents:

1) Programació de l'operació d'E/S

2) Transferència d'un bloc. Transferència d'una dada (per cada dada del bloc):

a) Sincronització
b) Intercanvi de la dada

3) Finalització de l'operació d'E/S



1.2.1. Programació de l'operació d'E/S

La programació de l'operació d'E/S és el procés per a indicar al mòdul d'E/S com s'ha de fer la transferència.

Aquesta programació consisteix a executar un petit conjunt d'instruccions que verifiquen si el perifèric està disponible per a iniciar una transferència de dades, actualitzen els registres del mòdul d'E/S, principalment els registres de

control per a donar instruccions al perifèric, i també ens pot servir per a inicialitzar les variables o registres que necessiti el processador per a dur a terme la transferència de dades.

1.2.2. Transferència de dades

La transferència de dades és la fase on es fa realment la transferència d'informació entre el processador i el mòdul d'E/S i és la fase que veurem en més detall analitzant diferents tècniques.

De manera genèrica en la transferència de cada dada dins d'un bloc distingim dues fases principals: la sincronització i l'intercanvi. Aquestes dues fases es repeteixen per cada dada del bloc.

La **sincronització** és on s'estableix un mecanisme per a aconseguir que el dispositiu més ràpid esperi que el dispositiu més lent estigui preparat per a dur a terme l'*intercanvi de la dada*.

Aquest mecanisme ens garanteix que no es deixin dades sense processar, però la conseqüència és que la transferència de dades és fa a la velocitat del dispositiu més lent. Generalment el dispositiu més lent és el perifèric, limitat per la seva pròpia capacitat de transferència i és la situació que dóna sentit a la gestió de l'E/S. Excepcionalment el processador pot ser més lent que el perifèric degut a que no es pot dedicar exclusivament a atendre a un únic perifèric.

Durant la fase de sincronització, el processador (o l'element que controli la transferència) ha de ser capaç de detectar quan està disponible el perifèric per a fer l'intercanvi d'una dada. El mòdul ha d'informar que el perifèric està preparat.

Exemple del procés de sincronització

Suposem que el processador envia deu dades (D0, D1, D2, D3, D4, D5, D6, D7, D8, D9) a un perifèric. El processador pot enviar una dada cada 2 ms i el perifèric triga 5 ms a processar cada dada.

	Temps (ms)	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Sense sincronització	Proc. envia	D0		D1		D2		D3		D4		D5		D6		D7		D8		D9		FI
	Perif. llegeix	↓ D0					↓ D2					↓ D5					↓ D7					↓ FI
Amb sincronització	Proc. envia	D0					D1					D2					D3					...
	Perif. llegeix	↓ D0					↓ D1					↓ D2					↓ D3					↓ ...

Quan no hi ha sincronització, observem que hi ha dades que es perden (D1, D3, D4, D6, D8, D9). Quan el perifèric vol llegir la dada següent, el processador ja n'ha enviat de noves i les anteriors s'han perdut perquè s'han sobreescrit en el registre de dades del mòdul d'E/S. Amb sincronització, el processador s'espera 3 ms (zona ombrejada) i, quan

el perifèric està preparat, el processador envia la nova dada. D'aquesta manera, no es perd cap dada, però la transferència es fa al ritme que marca el perifèric, que és el més lent.

Memòries intermèdies

Aquest problema es pot reduir utilitzant una petita memòria intermèdia (generalment anomenada *buffer*) per a emmagatzemar temporalment l'entrada o sortida de dades. Però si la diferència de velocitat és gran o els blocs de dades són grans, l'única manera de garantir que no es perdin dades és tenir un mecanisme de sincronització.

L'**intercanvi de la dada** és la fase en què s'envia realment la dada. Sabent que tant l'emissor com el receptor estan preparats (s'ha fet la *sincronització*), es verifica que la dada s'ha rebut correctament, es deixa tot preparat i s'indica que ja es pot iniciar una nova transferència.

En una operació d'entrada (lectura), el perifèric envia la dada al mòdul d'E/S, que fa una verificació per a detectar possibles errors. Generalment aquesta verificació és una validació molt simple, com és ara verificar el bit de paritat, i deixa la dada disponible per al processador en el registre de dades del mòdul d'E/S perquè el processador la pugui llegir.

En una operació de sortida (escriptura), el processador escriu la dada en el registre de dades del mòdul d'E/S i després el mòdul d'E/S l'envia al perifèric.

Vegem com seria una operació d'E/S en què el processador llegeix una dada d'un perifèric:

Nota

Cal tenir present que el procés que aquí descrivim pot variar lleugerament depenent del tipus de perifèric amb què fem la transferència, o de la tècnica d'E/S utilitzada.

1) El processador comprova l'estat del perifèric, llegeix el registre d'estat del mòdul d'E/S i mira els bits que indiquen si el perifèric està disponible. Pot ser necessari donar una ordre accedint al registre de control perquè el mòdul d'E/S consulti l'estat del perifèric i actualitzi el registre d'estat.

2) Si el perifèric està disponible, el processador sol·licita la dada enviant una ordre al mòdul d'E/S. El processador ha d'accedir al port que correspon al registre de control del mòdul d'E/S per a escriure-hi i així indicar l'operació que volem que faci. Hem programat la transferència.

Accés als ports d'E/S

Recordeu que per a accedir a un port d'E/S (registre d'un mòdul d'E/S) utilitzem instruccions de transferència si el processador té un mapa comú de memòria i E/S i instruccions específiques d'E/S si el computador té un mapa independent de memòria i E/S.

En aquestes instruccions hem d'especificar l'adreça d'un dels registres del mòdul d'E/S on tenim connectat el perifèric amb el qual volem dur a terme l'operació d'E/S.

El mòdul, per la seva part, ha de ser capaç de reconèixer aquesta adreça que el processador ha posat a les línies d'adreces del bus i llegir la dada de les línies de dades del bus per a actualitzar el registre corresponent si es tracta d'una operació d'escriptura o posar la informació d'aquest registre a les línies de dades del bus si estem fent una operació de lectura.

3) La lògica del mòdul s'encarrega d'identificar l'operació que ha de fer i inicia la transferència amb el perifèric de manera que el perifèric deixa d'estar disponible, i canvia l'estat a ocupat fins que la transferència finalitzi. Comença la transferència de la dada amb la sincronització.

4) El mòdul obté la dada del perifèric, la guarda al registre de dades perquè el processador la pugui llegir i actualitza el registre d'estat o avisa el processador per indicar que la dada està disponible, i aconsegueix d'aquesta manera la sincronització amb el processador. Si hi ha hagut errors en obtenir la dada del perifèric també ho indica en el registre d'estat. Mentre el mòdul d'E/S obté la dada del perifèric i valida la informació el processador s'ha d'esperar. S'acaba la sincronització i comença l'intercanvi de la dada.

5) Un cop la dada està preparada en el mòdul d'E/S i el processador n'està assabentat, el processador obté la dada, i es fa l'intercanvi de la dada. El processador ha d'accedir al port corresponent al registre d'estat si vol validar que la dada rebuda és correcta i al port corresponent al registre de dades per a llegir la dada. S'acaba l'intercanvi de la dada.

6) Com que només s'ha de transferir una dada, s'indica que l'estat del perifèric és de disponible per a fer una nova operació d'E/S. S'acaba l'operació d'E/S.

1.2.3. Finalització de l'operació d'E/S

La finalització de l'operació d'E/S és un procés semblant a la programació de l'operació d'E/S. També consisteix a executar un petit conjunt d'instruccions que actualitzen els registres del mòdul d'E/S, però ara per a indicar que s'ha finalitzat la transferència i que el mòdul d'E/S queda disponible per a atendre una altra operació d'E/S. També es pot consultar l'estat del mòdul d'E/S per a verificar que la transferència s'ha fet correctament.

Cal remarcar que tant la programació com la finalització de l'operació d'E/S són tasques que sempre fa el processador, ja que és qui sol·licita l'operació d'E/S d'informació, i tenen realment sentit quan volem enviar un bloc de dades i el mòdul d'E/S té autonomia per a gestionar la transferència. En canvi, no tenen tant sentit per a enviar dades unitàries. La complexitat d'aquest procés depèn de la complexitat del perifèric i de la tècnica d'E/S que utilitzem.

1.3. Gestió de múltiples dispositius

Tots els computadors han de gestionar més d'un perifèric i aquests poden treballar al mateix temps; per exemple, estem imprimint un document que tenim guardat al disc mentre escrivim un text amb el teclat que es mostra per pantalla. Per tant, hem de preveure que el nostre sistema d'E/S pugui gestionar transferències d'E/S amb dos perifèrics simultàniament o més. Això vol dir que de manera simultània dos mòduls d'E/S o més han d'estar preparats per a fer la transferència de dades amb el processador. La transferència no la podem fer al

mateix temps. Per aquest motiu hem de disposar d'un sistema que, primer, ens permeti determinar quins són els mòduls que hem d'atendre (**identificar els mòduls d'E/S** que estan preparats per a l'operació d'E/S) i, segon, ens permeti decidir qui atenem primer, tenint en compte que si ja atenem un altre perifèric o fem una altra tasca més prioritària no la podem interrompre (**establir una política de prioritats**).

Tant la identificació del mòdul d'E/S que està preparat per a transferir una dada com la manera d'establir una política de prioritats depenen de la tècnica d'E/S que utilitzem per a gestionar les operacions d'E/S i l'hem d'analitzar en detall en cada cas.

Les polítiques de prioritats bàsiques són:

- **Prioritat fixa**: quan el processador està preparat per a atendre una petició, sempre comença la consulta pel perifèric que té més prioritat. El perifèric amb més prioritat sempre és atès el primer. Si hi ha moltes peticions de perifèrics prioritaris, els menys prioritaris pot ser que s'esperin molt temps per a ser atesos.

- **Prioritat rotativa**: quan el processador està preparat per a atendre una petició, consulta el perifèric que hi ha a continuació segons un número d'ordre preestablert. Als perifèrics s'hi assigna un ordre però tots tenen la mateixa prioritat.

Normalment les polítiques de prioritat es defineixen segons les necessitats de cada dispositiu i les restriccions especifiques que molts cops imposa el sistema d'E/S utilitzat en un computador. I es pot utilitzar diferents polítiques de prioritats en un mateix sistema.

1.4. Tècniques d'E/S

Hem vist de manera genèrica quins són els passos per a fer una operació d'E/S. El primer pas i l'últim, la programació i la finalització de l'operació d'E/S, sempre són responsabilitat del processador i la complexitat que tenen depèn en forta mesura de les característiques del perifèric i del sistema d'interconnexió d'E/S que utilitzem, i no tant, tot i que també en depèn, de la tècnica d'E/S que utilitzem.

Per aquest motiu, en els apartats següents ens centrarem en la fase de **transferència d'una dada** (sincronització i intercanvi de la dada). Analitzarem quins dispositius ens permeten alliberar o descarregar al processador de les diferents tasques que s'han de fer en aquest procés i distingirem les tècniques bàsiques d'E/S següents:

- E/S programada.
- E/S per interrupcions.

- E/S per DMA.
- Canals d'E/S.

En la taula que hi ha a continuació veiem com queden repartides les responsabilitats en una transferència d'E/S segons la tècnica d'E/S que utilitzem.

	Programació	**Sincronització**	**Intercanvi**	**Finalització**
E/S programada	Processador	Processador	Processador	Processador
E/S per interrupcions	Processador	Mòdul d'E/S	Processador	Processador
E/S per DMA	Processador	DMA	DMA bloqueja el processador	Processador
Canals d'E/S	Processador / Canal d'E/S	Canal d'E/S	Canal d'E/S bloqueja el processador	Processador

2. E/S programada

Per a fer l'operació d'E/S entre el processador i el mòdul d'E/S, el processador executa un programa que controla tota l'operació d'E/S (programació, transferència de dades i finalització).

Analitzem en més detall la transferència d'una dada:

1) Sincronització. Durant la sincronització, el processador, com a responsable de la transferència, executa un programa que mira constantment l'estat del perifèric consultant el registre d'estat del mòdul d'E/S. Aquest programa té un bucle que s'executa contínuament fins que detecta el canvi d'estat i indica que el perifèric està preparat. Aquest mètode de sincronització s'anomena **sincronització per enquesta** o **espera activa**.

Mentre es fa la sincronització, el processador està dedicat al cent per cent a aquesta tasca i, per tant, no pot atendre altres processos o aplicacions. Si aquesta espera és molt llarga pot degradar el nivell de prestacions de tot el sistema. Per tant, és recomanable que les transferències fetes utilitzant aquesta tècnica siguin curtes i ràpides.

2) Intercanvi de la dada. Durant l'intercanvi de la dada, si és una operació de lectura (entrada) el processador llegeix el registre de dades del mòdul d'E/S per recollir la dada enviada pel perifèric, i la guarda a memòria; si és una operació d'escriptura (sortida) el processador agafa de la memòria la dada que volem enviar al perifèric i l'escriu en el registre de dades del mòdul d'E/S.

Programa per a atendre un perifèric utilitzant E/S programada

Volem enviar una dada (escriptura) a un perifèric controlat per un mòdul d'E/S que té mapats els registres de control, d'estat i de dades a les adreces 0120h, 0124h, 0128h, respectivament.

SINCRO:	**IN**	R0,[0000 0124h]	0000 0124h: adreça del registre d'estat del mòdul d'E/S. Llegim l'estat i el guardem a R0.
	AND	R0, 00000008h	00000008h és una màscara per a mirar el bit 3 del registre R0; en aquest cas és el bit que indica si el perifèric està disponible per a fer la transferència de dades. (El primer bit, bit menys significatiu, és el bit 0.)
	JE	SINCRO	Si el bit 3 és a zero tornem a SINCRO: i continuem esperant fins que el perifèric estigui preparat per a rebre la dada i el mòdul activi aquest bit; si val 1 continuem i anem a INTERCANVI: per fer l'intercanvi de la dada.
INTERCANVI:	**MOV**	R1, [Dada]	Agafem la dada que volem enviar al perifèric i la posem en el registre R1.
	OUT	[0000 0128h], R1	Copiem la dada que tenim en el registre R1 al registre de dades del mòdul d'E/S (0000 0128h) perquè aquest l'enviï al perifèric.
	RET		Retornem el control al programa que ha cridat a aquesta rutina per fer l'operació d'E/S.

Tant aquest exemple com la descripció del procés de sincronització i transferència de dades s'ha fet seguint un model simplificat d'operació d'E/S. Cal tenir present que per a programar una rutina que dugui a terme una operació d'E/S amb un perifèric concret cal conèixer les adreces dels ports d'E/S associades als registres del mòdul d'E/S que gestiona aquest perifèric i el significat o utilitat de cadascun dels bits. Atesa la gran varietat de perifèrics i la creixent complexitat, interpretar tota aquesta informació és una tasca força complicada i generalment són els fabricants mateixos qui desenvolupen aquestes rutines.

2.1. Gestió de múltiples dispositius

No és habitual gestionar operacions d'E/S amb múltiples dispositius utilitzant E/S programada. En cas de ser necessari cal tenir present que durant la sincronització s'ha de fer l'enquesta de tots els perifèrics implicats establint una política de prioritats implementada en forma de programa durant la sincronització. Aquest sistema d'identificar quin perifèric necessita ser atès s'anomena **enquesta** o ***polling***.

En el moment en què es detecta que un dels perifèrics està preparat cridem a una subrutina per a fer l'intercanvi de la dada i retornem al bucle de sincronització per seguir fent l'enquesta segons la política de prioritats implementada fins que s'hagin acabat totes les operacions d'E/S que estan programades.

3. E/S amb interrupcions

En aquest apartat veurem l'E/S per interrupcions. Aquesta tècnica d'E/S pretén evitar que el processador hagi d'estar aturat o fent feina improductiva mentre espera que el perifèric estigui preparat per a fer una nova operació d'E/S i pugui aprofitar aquest temps per a executar altres programes.

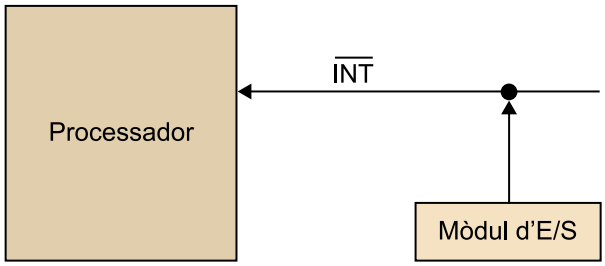
Usant la tècnica d'E/S per interrupcions la sincronització entre el perifèric i el processador és duta a terme pel mòdul d'E/S, de manera que es descarrega el processador d'aquesta responsabilitat.

Per a utilitzar aquesta tècnica d'E/S en un computador, però, és necessari considerar tant aspectes del programari com del maquinari.

Com a part del maquinari, cal que el computador disposi d'una línia especial que ha de formar part del conjunt de línies de control del bus del sistema i que anomenem **línia de petició d'interrupció (INT)**. El mòdul d'E/S avisa el processador mitjançant aquesta línia, i indica que està preparat per a fer la transferència. El senyal INT l'activa el mòdul d'E/S i el rep el processador. És un senyal actiu a la baixa. El processador ha de tenir un punt de connexió d'entrada per on arribaran les interrupcions i el mòdul d'E/S ha de tenir un punt de connexió de sortida per on generarà les interrupcions.

Senyal actiu a la baixa

El senyal INT és actiu a la baixa. Es considera actiu si tenim un 0 a la línia i no es considera actiu si hi tenim un 1.



Per a fer una operació d'E/S utilitzant aquesta tècnica se segueixen els mateixos passos que l'E/S programada: es programa l'operació d'E/S, es fa la transferència de dades i es finalitza l'operació d'E/S. La diferència principal la tenim durant la transferència de dades, en què en la fase de sincronització hem de fer la gestió de les interrupcions i això també afecta en certa mesura l'intercanvi de dades, com veurem més endavant.

Durant la fase de sincronització, un cop feta la programació de l'operació d'E/S, el processador executa un altre programa (segons la política de gestió de processos del sistema operatiu) de manera que el processador estarà ocupat fent feina productiva fins que el mòdul d'E/S estigui preparat i activi el senyal de petició d'interrupció (INT).

D'entrada, el processador no sap en quin moment es produirà aquesta petició; per tant, ha de comprovar periòdicament si el mòdul d'E/S demana l'atenció del processador, sense que això afecti la dedicació que hi té. Aquesta comprovació el processador la fa dins del cicle d'execució de cada instrucció. És una operació molt ràpida que fins i tot es pot encavalcar amb el començament de la lectura de la instrucció següent perquè no afecti el rendiment del processador. Els processadors que han de gestionar interrupcions han de tenir en el cicle d'execució d'instrucció una **fase de comprovació d'interrupcions**.

Cicle d'execució d'instrucció

Recordeu que les quatre fases principals per a executar una instrucció són:

1) Lectura de la instrucció.
2) Lectura dels operands font.
3) Execució de la instrucció i emmagatzematge de l'operand destinació.
4) Comprovació d'interrupcions.

En el moment en què el processador reconeix que ha arribat una petició d'interrupció, comença un **cicle de reconeixement d'interrupció** per a aturar l'execució del programa actual i transferir el control a la **rutina de servei de la interrupció (RSI)**, rutina que accedeix al mòdul d'E/S corresponent per dur a terme la transferència de dades i un cop s'acabi l'execució de l'RSI continuar l'execució del programa que havíem aturat fent el **retorn d'interrupció**.

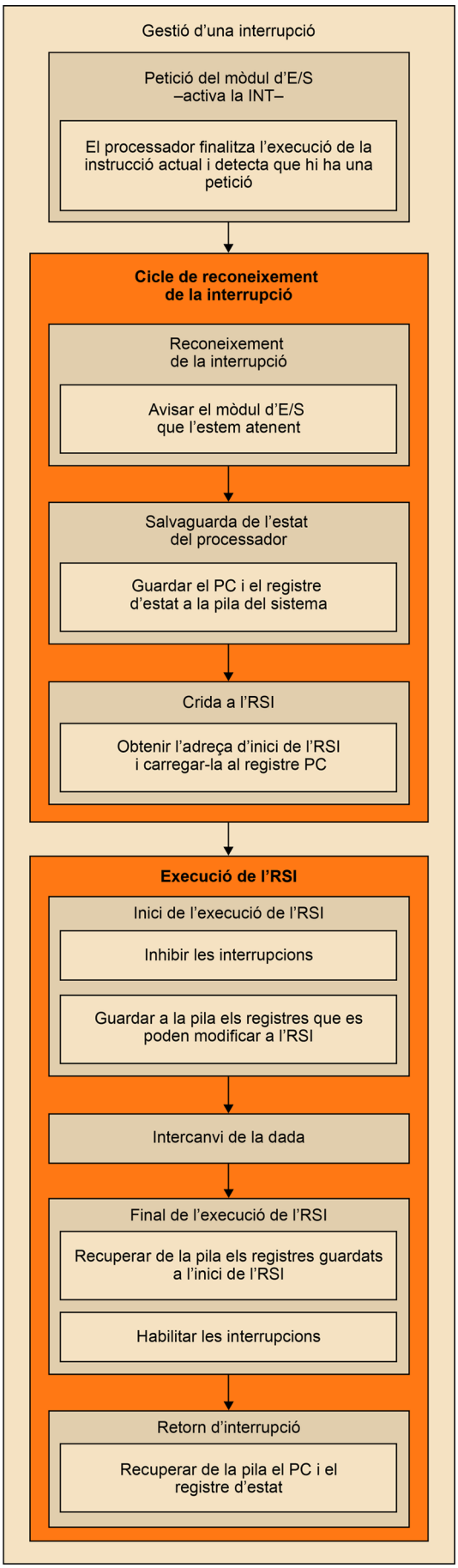
En aquest apartat ens centrarem en l'anàlisi d'interrupcions provocades per elements externs al processador (interrupcions del maquinari), però també hi ha un tipus d'interrupcions produïdes per esdeveniments interns al processador anomenades **interrupcions de programari**, **excepcions** o **traps**.

Les interrupcions del programari es gestionen d'una manera molt semblant a les interrupcions externes al processador però sense ser necessari dedicar una línia del bus del sistema per a gestionar-les. Per a tractar-les no cal esperar la fase de comprovació d'interrupció, sinó que es pot tractar en el mateix moment en què es produeix.

Les més habituals són aquestes: operacions que l'ALU no pot fer, com dividir per 0, no es reconeix el codi d'operació de la instrucció que volem executar, accés a recursos restringits o àrees privilegiades de memòria i també peticions del programador mateix amb instruccions específiques del joc d'instruccions per a fer peticions al sistema operatiu.

3.1. Gestió d'una interrupció amb un únic mòdul d'E/S

Vegem quins són els passos bàsics per a la gestió d'una interrupció en un sistema amb una única línia d'interrupció i un únic mòdul d'E/S. Més endavant analitzarem diferents millores d'aquest sistema per a gestionar múltiples mòduls d'E/S i múltiples línies d'interrupció i com afecta això aquest procés. Primer, però, cal entendre bé el cas més simple.


Gestió d'una interrupció
Petició del mòdul d'E/S
–activa la INT–
El processador finalitza l'execució de la instrucció actual i detecta que hi ha una petició
Cicle de reconeixement de la interrupció
Reconeixement de la interrupció
Avisar el mòdul d'E/S que l'estem atenent
Salvaguarda de l'estat del processador
Guardar el PC i el registre d'estat a la pila del sistema
Crida a l'RSI
Obtenir l'adreça d'inici de l'RSI i carregar-la al registre PC
Execució de l'RSI
Inici de l'execució de l'RSI
Inhibir les interrupcions
Guardar a la pila els registres que es poden modificar a l'RSI
Intercanvi de la dada
Final de l'execució de l'RSI
Recuperar de la pila els registres guardats a l'inici de l'RSI
Habilitar les interrupcions
Retorn d'interrupció
Recuperar de la pila el PC i el registre d'estat

Un cop el processador ha sol·licitat una operació d'E/S, ha programat la transferència i ja executa un altre programa mentre espera que el mòdul d'E/S estigui preparat. En el moment en què el mòdul d'E/S demana l'atenció del processador (el mòdul activa la INT), es produeix una seqüència d'esdeveniments que el processador ha de gestionar per a atendre aquesta petició del mòdul d'E/S, garantint que després podrà tornar el control al programa al qual aturem l'execució per atendre la petició del mòdul d'E/S.

Els passos són els següents:

1) Petició del mòdul d'E/S. El mòdul d'E/S està preparat per a fer una transferència i activa la INT. Aleshores, quan el processador acaba l'execució de la instrucció actual, en la darrera fase del cicle d'execució de la instrucció, la fase de comprovació d'interrupció, detecta que s'ha fet una petició.

2) Cicle de reconeixement de la interrupció. Aquesta és segurament la part més complexa de la gestió d'interrupcions perquè es produeixen molts esdeveniments en poc temps i com veurem més endavant alguns d'aquests esdeveniments es poden produir encavalcadament en el temps i cal estar atent a qui genera l'acció, qui la rep i quina resposta hi dóna.

a) Reconeixement de la interrupció. Si les interrupcions estan habilitades, el processador accepta la petició (quan hi ha més d'un dispositiu s'ha de determinar si és prou prioritari per a ser atès, situació que analitzarem més endavant ja que té altres implicacions en el procés). Aleshores cal que el processador avisi el mòdul d'E/S perquè sàpiga que l'està atenent i perquè desactivi la INT, i també cal que el processador inhibeixi les interrupcions per evitar noves peticions.

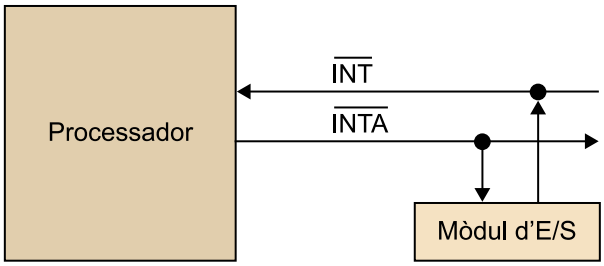
Si les interrupcions estan inhibides el processador no atén la petició i continua executant el programa. El perifèric s'ha d'esperar a ser atès i ha de deixar la INT activa fins que el processador les torni a habilitar.

Per a **inhibir les interrupcions**, hi ha bàsicament dues maneres de fer-ho: amb un maquinari específic que les inhibeixi automàticament fins que s'acabi l'atenció de la interrupció o deixar que sigui responsabilitat de la mateixa RSI, que ha d'executar una primera instrucció per a inhibir les interrupcions i en acabar executar una altra instrucció per a tornar-les a habilitar.

Inhibir les interrupcions consisteix a modificar un o més bits del registre de la paraula d'estat del processador (alguns processadors poden tenir registres específics per a la gestió d'interrupcions). Aquests bits són els que es consulten en la fase de comprovació d'interrupcions durant l'execució d'una instrucció per a saber si les interrupcions estan habilitades i s'ha d'acceptar o no una petició.

Si el processador no inhibeix les interrupcions, en executar la primera instrucció de l'RSI, en la fase de comprovació d'interrupció tornarà a acceptar la petició i podria entrar en un bucle infinit executant aquesta primera instrucció indefinidament, ja que tornarà a començar un cicle de reconeixement de la mateixa interrupció.

Anàlogament a la manera com es fa en la inhibició de les interrupcions, es pot avisar el mòdul d'E/S de dues maneres: la més habitual és utilitzant un maquinari específic que inclou una **línia de reconeixement d'interrupció** (INTA o INTACK, de l'anglès *interrupt acknowlege*) que activa el processador, igual que la INT és un senyal actiu a la baixa, o deixar que sigui responsabilitat de la mateixa RSI, que ha d'executar una o diverses instruccions per a accedir als registres del mòdul d'E/S i d'aquesta manera indicar-li que ja l'atén.



b) Salvaguarda de l'estat del processador. Ara el processador ha reconegut que hi ha una petició d'un mòdul d'E/S, l'ha acceptat i n'ha d'executar la rutina de servei d'interrupció (RSI) per a atendre'l, però recordem que el processador està executant un altre programa i cal reprendre'n l'execució un cop ha acabat l'execució de l'RSI. Per aquest motiu s'ha de fer la salvaguarda de l'estat del processador, que consisteix a emmagatzemar la informació necessària per a garantir que podrem reprendre l'execució del programa en les mateixes condicions. Aquesta informació s'emmagatzema a la memòria del computador, generalment a la pila de sistema.

L'estat del processador el determina el valor de tots els seus registres, però per a reprendre l'execució del programa només cal guardar els registres que pugui modificar l'RSI. Una RSI pot modificar qualsevol registre del processador, però realment solen ser rutines relativament simples que només modifiquen alguns d'aquests registres. Si només es guarden els registres que es modifiquen dins l'RSI, la salvaguarda i la restauració posterior serà molt més eficient en temps i espai, ja que reduïm els accessos a la memòria i l'espai necessari per a guardar l'estat.

La manera més habitual de fer la salvaguarda és emmagatzemar de manera automàtica la informació mínima necessària per a cridar a l'RSI i emmagatzemar la resta d'informació dins la mateixa rutina i és responsabilitat del programador de l'RSI determinar quins registres cal guardar.

La informació mínima que s'ha de guardar és el **registre comptador de programa** (PC) i el **registre d'estat**. El registre PC té l'adreça de la instrucció següent que volem executar. El registre d'estat conté informació important de control i d'estat del processador que pot ser modificada per qualsevol instrucció.

Nota

Recordeu que el valor del PC s'actualitza en la fase de lectura de la instrucció del cicle d'execució.

Pel que fa a la resta d'informació, com que el processador no pot saber quins registres modificarà l'RSI, deixa que sigui responsabilitat de la mateixa RSI.

c) Crida a l'RSI. Per a començar a executar la rutina de servei d'interrupció (RSI), primer de tot, cal saber l'adreça de memòria on hem d'iniciar l'execució de l'RSI. En cas de gestionar un únic dispositiu aquesta adreça pot ser fixa (quan tenim més d'un dispositiu, en la fase de reconeixement de la interrupció hem d'haver identificat quin dispositiu demana atenció i utilitzarem aquesta informació per a saber l'adreça de l'RSI corresponent al dispositiu que hem d'atendre).

Un cop sabem l'adreça d'inici de l'RSI es carrega en el registre PC i s'inicia el cicle d'execució de la primera instrucció de l'RSI. L'RSI s'executa en modalitat supervisor.

3) Execució de la rutina de servei d'interrupció

a) Inici de l'execució de l'RSI. Si no s'ha fet la inhibició de les interrupcions en la fase de reconeixement de la interrupció cal fer-ho a l'inici de l'RSI. Aquesta inhibició es fa utilitzant una instrucció específica del joc d'instruccions.

Després s'han de guardar els registres que puguin ser modificats a la pila (el programador de l'RSI sap quins registres es modifiquen i, per tant, pot determinar quins registres s'han de guardar) i abans d'acabar l'execució cal restaurar-los amb els valors originals. Quan es programa una RSI s'ha de ser molt curós amb la salvaguarda dels registres perquè pot provocar que altres programes (els que aturem per executar l'RSI) funcionin malament.

b) Intercanvi de la dada. La part principal del codi de l'RSI ha de ser específic per al perifèric que atenem. De manera general s'ha d'accedir al registre de dades per a fer l'intercanvi de la dada. Si és necessari, es pot accedir primer als registres d'estat i de control per saber si l'operació d'E/S s'ha fet correctament o s'han produït errors.

Per a programar el codi d'una RSI cal conèixer en detall el funcionament del perifèric i quina funció té cadascun dels bits dels registres del mòdul d'E/S que s'utilitza per a controlar-lo. Generalment aquesta informació que cal conèixer és força complexa quan s'ha d'analitzar en detall i pot variar molt d'un dispositiu a un altre, fins i tot, en evolucions d'un mateix tipus de dispositiu.

c) Finalització de l'execució de l'RSI. Cal recuperar el valor dels registres del processador que s'han guardat al començament de l'RSI a la pila (en ordre invers a la manera com s'ha guardat ja que ho tenim guardat a la pila).

Si s'han inhibit les interrupcions al començament de l'RSI, cal habilitar-les utilitzant una instrucció específica del joc d'instruccions.

d) Retorn d'interrupció. Abans d'acabar l'execució de l'RSI hem de restaurar l'estat del processador per a tornar el control al programa que s'estava executant. Per això hem de recuperar de la pila del sistema la informació que s'ha guardat de manera automàtica (els registres d'estat i el PC). Aquesta informació la recuperem en executar la darrera instrucció de l'RSI, que és la instrucció de retorn d'interrupció habitualment anomenada *IRET*.

Un cop restaurat l'estat del processador, es reprèn l'execució del programa que hem aturat en les mateixes condicions iniciant un nou cicle d'execució de la instrucció. I la petició del mòdul d'E/S ha quedat atesa.

Exemple

En aquest exemple s'observa la manera com s'atén una interrupció. És especialment important fixar-se com queda el registre PC després d'executar cada fase i l'evolució de la pila durant aquest procés.

Les abreviatures que es fan servir són:

- SP: *stack pointer* (punter al cim de la pila).
- PC: *program counter* (adreça de la instrucció que estem executant).
- RSI: rutina de servei d'interrupció.
- INT: senyal que activa el mòdul d'E/S per demanar atenció al processador.


1. Petició del mòdul d'E/S. En acabar l'execució de la instrucció actual el processador mira si la INT és activa i passa a atendre la petició.
Rutina per a atendre una INT
RSI: Inhibir INT
PUSH Regs.
...
Intercanvi de la dada
...
POP Regs.
Habilitar INT
IRET
...
Pila del sistema
SP → Cim de la pila
...
Programa en execució
PC → Instrucció actual
Instrucció següent
...
arriba ← INT
2. Reconeixement de la interrupció, salvaguarda de l'estat (guarda PC i registre d'estat) i crida a l'RSI (actualitza PC amb l'adreça d'inici de l'RSI).
Rutina per a atendre una INT
PC → RSI Inhibir INT
PUSH Regs.
...
Intercanvi de la dada
...
POP Regs.
Habilitar INT
IRET
...
Pila del sistema
SP → PC (Inst. següent)
Reg. STATUS
Cim de la pila
...
Programa en execució
...
Instrucció actual
Instrucció següent
...
3. S'inicia l'execució de l'RSI (inhibir les interrupcions i guardar els registres que poden ser modificats per l'RSI).
Rutina per a atendre una INT
RSI: Inhibir INT
PUSH Regs.
PC → ...
Intercanvi de la dada
...
POP Regs.
Habilitar INT
IRET
...
Pila del sistema
SP → Regs. de l'RSI
PC (Inst. següent)
Reg. STATUS
Cim de la pila
...
Programa en execució
...
Instrucció actual
Instrucció següent
...
4. Intercanvi de la dada.
Rutina per a atendre una INT
RSI: Inhibir INT
PUSH Regs.
...
Intercanvi de la dada
...
PC → POP Regs.
Habilitar INT
IRET
...
Pila del sistema
SP → Reg. de la INT
PC (Instr. següent)
Reg. STATUS
Cim de la pila
...
Programa en execució
...
Instrucció actual
Instrucció següent
...
5. Final de l'execució de l'RSI. Recuperar de la pila els registres guardats i habilitar les interrupcions.
Rutina per a atendre una INT
RSI: Inhibir INT
PUSH Regs.
...
Intercanvi de la dada
...
POP Regs.
Habilitar INT
PC → IRET
...
Pila del sistema
SP → PC (Instr. següent)
Reg. STATUS
Cim de la pila
...
Programa en execució
...
Instrucció actual
Instrucció següent
...
6. Retorn d'interrupció. Recuperar el registre d'estat i el PC de la pila, i deixar el processador en el mateix estat i el PC apuntant a la instrucció següent.
Rutina per a atendre una INT
RSI: Inhibir INT
PUSH Regs.
...
Intercanvi de la dada
...
POP Regs.
Habilitar INT
IRET
...
Pila del sistema
SP → Cim de la pila
...
Programa en execució
...
Instrucció actual
PC → Instrucció següent
...

3.2. Gestió d'interrupcions amb múltiples mòduls d'E/S

Fins ara hem estudiat el cas més simple, en què només tenim un mòdul que funciona per interrupcions, però la realitat és que un sistema d'E/S per interrupcions ha de gestionar múltiples dispositius amb capacitat de generar interrupcions. Vegem com afecta això la gestió d'interrupcions i com ho hem d'implementar.

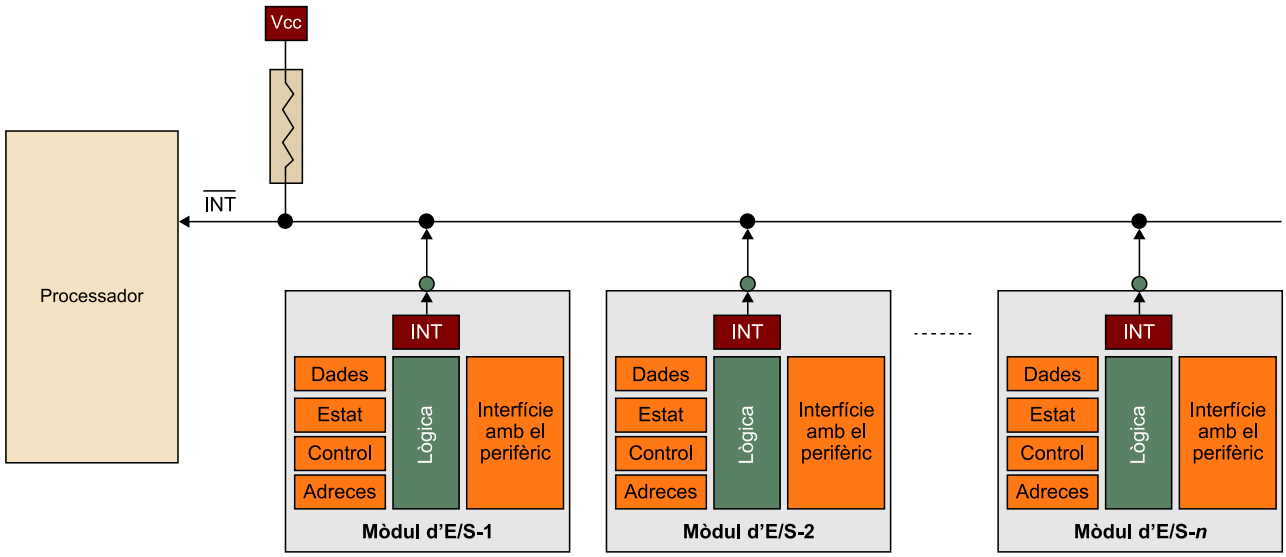
Quan el processador rep una petició, sap que algun mòdul d'E/S demana atenció però no sap quants demanen atenció, ni quins són. Per a identificar qui demana atenció i decidir qui s'ha d'atendre les qüestions principals que cal resoldre són:

- Identificar quin és el perifèric que demana atenció.
- Determinar quin perifèric hem d'atendre primer en cas que dos perifèrics o més sol·licitin atenció al mateix temps.
- Gestionar, si el sistema permet la nidificació, les prioritats per a determinar si el perifèric que sol·licita atenció és més prioritari que un altre que ja atenem.
- Obtenir l'adreça de l'RSI corresponent al perifèric que hem d'atendre.

Aquestes qüestions afecten principalment el cicle de reconeixement de la interrupció i fan aquest procés força més complex. L'objectiu principal dels sistemes que ens permeten gestionar més d'un mòdul d'E/S mitjançant interrupcions és aconseguir que aquest procés sigui tan eficient com sigui possible, afegint millores al maquinari que dóna suport al sistema d'atenció d'interrupcions.

3.3. Sistema amb una única línia de petició d'interrupció

Aquest és el cas més simple, en què tots els mòduls d'E/S es connecten en col·lector obert, utilitzant un PULL-UP, a una única línia de petició d'interrupció que arriba al processador.



La resistència connectada a Vcc (tensió alta) serveix per a implementar un PULL-UP, que dóna la funcionalitat d'OR-CABLEJADA considerant que la línia és activa a la baixa. Si tenim un 0 a la línia es considera activa i si hi tenim un 1 no es considera activa. Això permet que els mòduls d'E/S estiguin connectats en col·lector obert a la línia de manera que si cap mòdul n'activa la sortida el *pull-up* manté la línia a 1 (tensió alta), condició de repòs de tots els dispositius, i si algun mòdul activa la sortida, posant un 0 (tensió baixa), la línia canvia d'estat i el processador pot reconèixer que hi ha una petició pendent de ser atesa.

La gestió d'una interrupció en aquest sistema és anàloga a la gestió d'una interrupció amb un únic mòdul d'E/S. Ara, però, per a identificar quin perifèric demana atenció i gestionar les prioritats si més d'un mòdul d'E/S ha demanat atenció al mateix temps, el processador executa una única RSI que té una adreça d'inici fixa. L'RSI, mitjançant codi i accedint als registres d'estat de cada mòdul d'E/S, determina el mòdul d'E/S que ha d'atendre, de manera molt semblant a l'enquesta (o *polling*) que es fa en E/S programada quan tenim més d'un mòdul d'E/S. El codi que utilitzem per a atendre el perifèric és una subrutina de l'RSI.

Els avantatges principals que ofereix aquest sistema són:

- Com que fem l'enquesta de tots els mòduls per programa, és molt flexible determinar les prioritats perquè podem implementar diferents polítiques de prioritats només modificant el codi de l'RSI.

- No cal fer canvis en el maquinari.

Els desavantatges principals són:

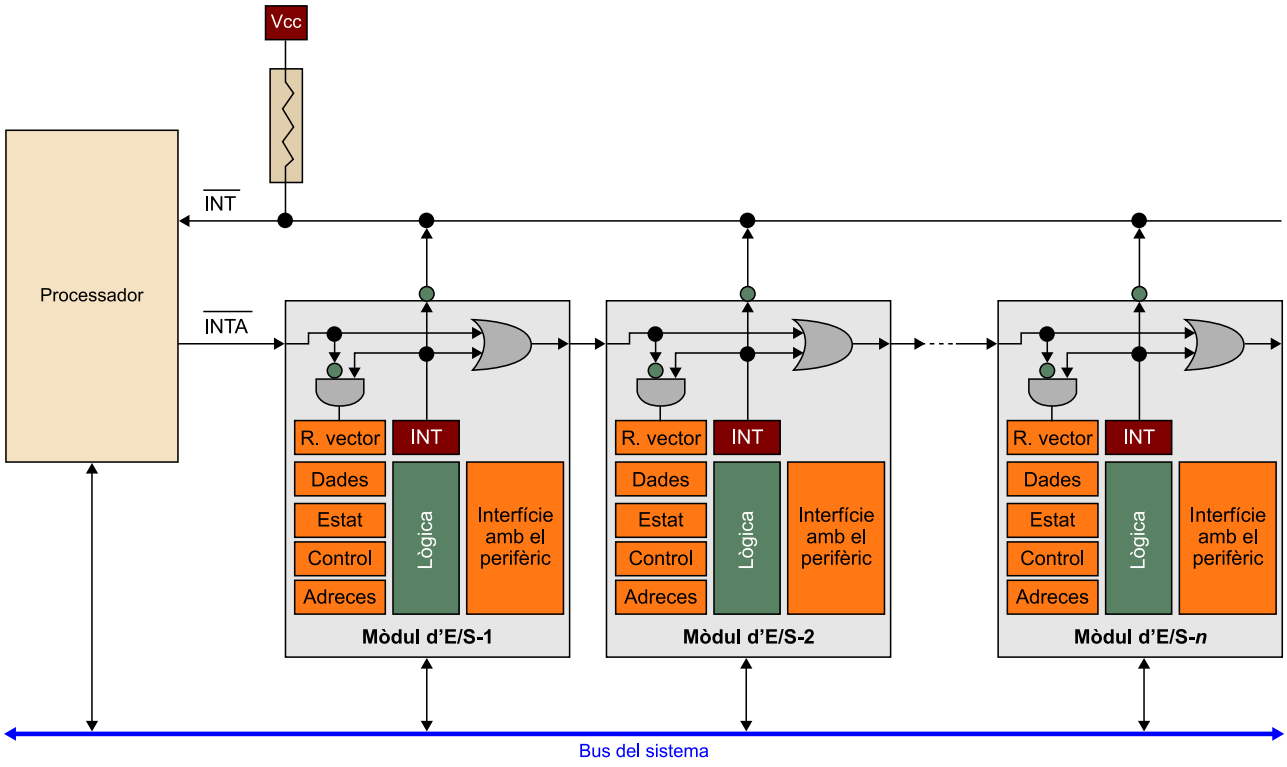
- L'execució del codi per a fer l'enquesta dels mòduls i la gestió de prioritats és molt costosa en temps ja que estem executant un petit programa que accedeix a tots els mòduls d'E/S connectats a la línia.

- Com que hi ha una sola línia de petició d'interrupció, s'ha d'inhibir les interrupcions i, per tant, mentre atenem una petició no podem atendre'n d'altres encara que siguin més prioritàries.

3.4. Sistema amb una línia de petició d'interrupció i una línia de reconeixement amb encadenament

Aquest sistema amb encadenament també s'anomena ***daisy-chain***. Els mòduls d'E/S es connecten al processador amb una línia de petició d'interrupció (INT) en col·lector obert i una línia de reconeixement d'interrupció (INTA) que genera el processador per indicar al mòdul d'E/S que s'atén la petició que ha fet. Aquest senyal INTA es propaga mitjançant els mòduls.

És un sistema de connexió que permet, per una banda, reduir el temps necessari per a gestionar les prioritats si més d'un mòdul d'E/S ha demanat atenció al mateix temps i, per altra banda, identificar quin perifèric demana atenció, sense haver d'executar un programa d'enquesta que accedeix a tots els mòduls d'E/S.



Per a implementar aquest sistema cal fer alguns canvis en el maquinari: el processador necessita disposar d'un punt de connexió per on generar el senyal de reconeixement d'interrupció (INTA); els mòduls d'E/S han de disposar d'un circuit lògic molt simple per a propagar el senyal INTA al següent mòdul d'E/S deixant tots els mòduls d'E/S connectats un darrere l'altre i disposar d'un nou registre que anomenem **registre vector** on emmagatzemem un valor anomenat **vector d'interrupció** que enviem al processador mitjançant el bus del sistema. Aquest valor que emmagatzema cada registre vector ha de ser únic, ja que el processador l'utilitza per a identificar el perifèric que demana atenció.

La gestió d'una interrupció en aquest sistema és anàloga a la gestió d'una interrupció amb un únic mòdul d'E/S, i varia el cicle de reconeixement de la interrupció per a gestionar les prioritats si més d'un mòdul d'E/S ha demanat atenció al mateix temps i identificar quin perifèric demana atenció.

3.4.1. Interrupcions vectoritzades

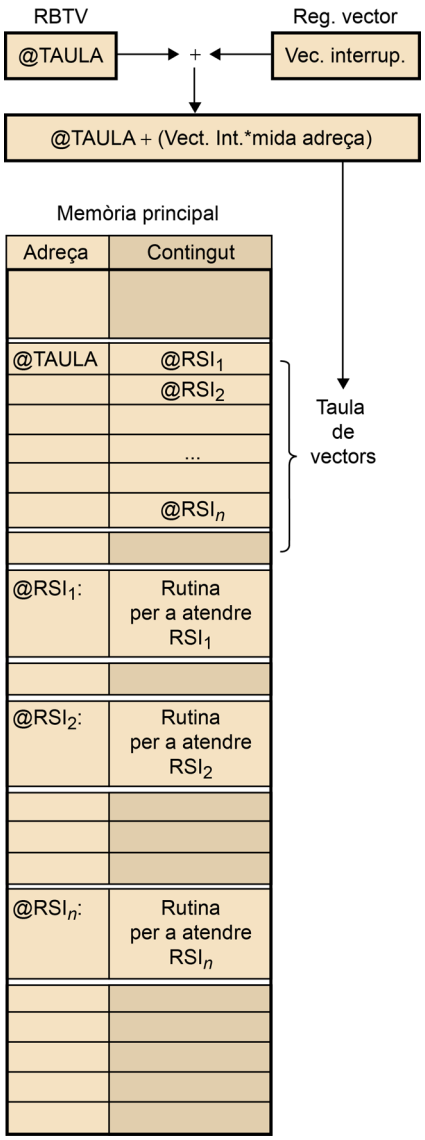
S'anomena **vectorització** la tècnica en què el processador identifica el mòdul d'E/S mitjançant la informació que envia el mateix mòdul d'E/S. Quan per a tractar les interrupcions utilitzem aquesta tècnica per a identificar qui fa la petició diem que tenim les **interrupcions vectoritzades**.

Un cop un mòdul d'E/S o més d'un han fet la petició al processador activant la INT i el processador accepta la petició, activa l'INTA i comença el procés per a saber quin s'ha d'atendre.

Tractament del senyal INTA

El senyal INTA té un funcionament semblant al senyal READ per a llegir una dada de memòria. Quan el mòdul rep el senyal INTA i té la INT activa, diposita el vector d'interrupció al bus del sistema perquè el processador el pugui llegir.

Aquest senyal INTA provinent del processador, actiu a la baixa, arriba al primer mòdul. Si aquest mòdul ha sol·licitat atenció bloqueja la propagació del senyal INTA i diposita al bus del sistema el vector d'interrupció, valor emmagatzemat al registre vector del mòdul d'E/S; si no ha sol·licitat atenció (no té activa la INT) deixa passar el senyal INTA al mòdul següent, que fa el mateix fins que arriba al darrer mòdul connectat a aquesta línia.



Com veiem, els mòduls d'E/S queden encadenats per aquest senyal INTA. El primer mòdul de la cadena és el primer de rebre el senyal i, per tant, el més prioritari, i el propaga fins al darrer, que és el menys prioritari. Si volem canviar les prioritats ens veurem obligats a canviar l'ordre dels mòduls dins la cadena físicament.

El processador llegeix el vector d'interrupció del bus del sistema. Amb aquest valor identifica el perifèric que ha d'atendre i l'utilitza per a obtenir l'adreça d'inici de l'RSI (aquesta adreça és la que hem de carregar al registre PC per a fer la crida a l'RSI).

La manera més habitual d'obtenir l'adreça d'inici de l'RSI és utilitzar una **taula de vectors d'interrupció** emmagatzemada a la memòria principal, on tenim guardades les adreces d'inici de les RSI associades a cada petició d'interrupció que hem d'atendre. El vector d'interrupció l'utilitzem per a accedir a la taula. Aquesta taula pot estar emmagatzemada en una adreça fixa de memòria o en una adreça programable emmagatzemada en el **registre base de la taula de vectors (RBTV)**. Si tenim la taula de vectors en una adreça fixa, hem de deter-

minar la posició dins de la taula de vectors, a partir del vector d'interrupció. Si tenim la taula de vectors en una adreça programable, hem de determinar la posició dins la taula de vectors sumant l'RBTV a un índex obtingut a partir del vector d'interrupció.

Desplaçament dins la taula de vectors

Una manera d'obtenir el desplaçament dins la taula de vectors a partir del vector d'interrupció, com que el vector d'interrupció identifica el dispositiu (i no l'adreça dins la taula o l'adreça de la rutina mateixa), és multiplicar el vector d'interrupció per les posicions de memòria que ocupa una adreça dins la taula de vectors. Si les adreces de memòria emmagatzemades a la taula de vectors ocupen 4 bytes (32 bits) i cada posició de memòria és d'un byte, el vector d'interrupció s'ha de multiplicar per 4. Fer aquesta operació en binari és molt simple: només cal desplaçar el vector d'interrupció dues posicions a l'esquerra (afegir dos zeros a la part dreta) per a obtenir el desplaçament dins la taula. En sistemes més complexos pot ser necessari fer un altre tipus d'operacions.

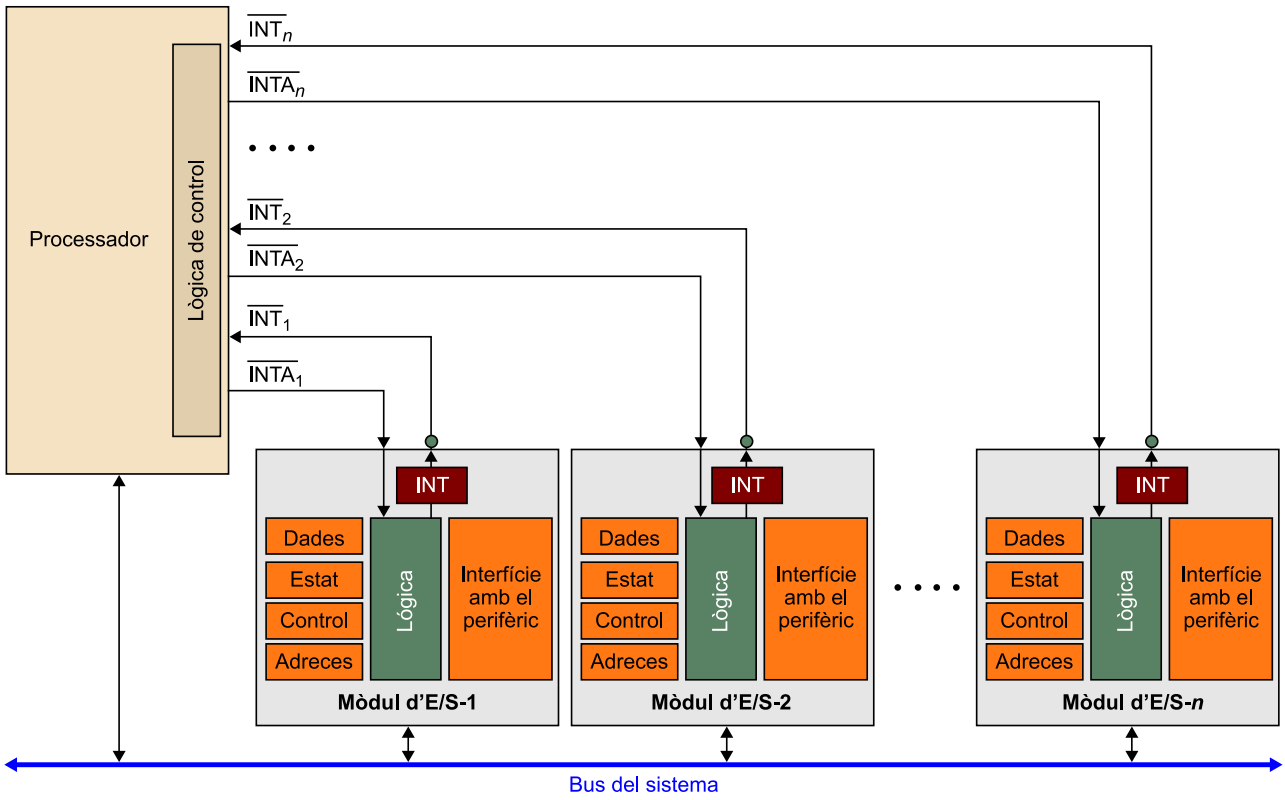
Els principals avantatges d'aquest sistema amb encadenament són que la identificació és fa en un temps semblant a la duració d'un cicle de bus, és molt menys costós en temps que fer l'enquesta a tots els mòduls d'E/S i també resol la gestió de prioritat quan hi ha peticions simultànies.

Els desavantatges principals són que el sistema de prioritats és fix i inalterable –canviar l'ordre de prioritats implica modificar el maquinari– i que, com que té una única línia de petició d'interrupcions, no permet que altres perifèrics més prioritaris siguin atesos si ja atenem una petició d'interrupció sigui quina sigui la prioritat que té.

3.5. Sistema amb línies independents de petició d'interrupcions i de reconeixement

En aquest sistema tenim una línia de petició d'interrupció i una de reconeixement d'interrupció per cada mòdul d'E/S. D'aquesta manera aconseguim que la identificació del perifèric sigui immediata i permetem la **nidificació d'interrupcions**, és a dir, una interrupció pot interrompre l'execució d'una RSI que atén una petició menys prioritària que s'ha produït amb anterioritat.

Per a implementar aquest sistema cal que el processador disposi dels punts de connexió necessaris per a connectar els senyals de petició i reconeixement de cadascun dels mòduls d'E/S que puguin generar una interrupció cap al processador i un circuit específic per a fer el control d'aquestes línies de petició i reconeixement d'interrupcions. Els mòduls d'E/S només és necessari que puguin generar una petició d'interrupció i rebre el senyal de reconeixement d'interrupció generat pel processador.



La gestió d'una interrupció en aquest sistema és anàloga a la gestió d'una interrupció amb un únic mòdul d'E/S, i varia la fase de reconeixement de la interrupció per a gestionar les prioritats i identificar quin perifèric demana atenció.

El procés per a fer el cicle de reconeixement de la interrupció és el següent: un cop un o més mòduls d'E/S han fet la petició al processador activant la INT, el processador decideix si accepta la petició, inhibeix de manera selectiva les interrupcions, activa l'INTA corresponent i obté l'adreça de l'RSI per atendre aquella petició.

En un sistema amb una única línia de petició d'interrupció hem vist que quan acceptem una petició cal inhibir les interrupcions i, per tant, no podem acceptar noves peticions fins que s'ha acabat l'atenció d'aquesta petició. En un sistema amb múltiples línies de petició i reconeixement d'interrupció també hem d'inhibir les peticions d'interrupció de la línia de què hem acceptat la petició, però cal tenir un mecanisme per a decidir si s'ha de permetre peticions de les altres línies per a emmascarar les interrupcions de manera selectiva segons una política de prioritats.

Hi ha dues formes bàsiques per a emmascarar de manera selectiva les interrupcions:

- Emmascarament individual.
- Emmascarament per nivell.

En l'**emmascarament individual** disposem d'un **registre d'interrupcions**, que té 2 bits per cada línia de petició d'interrupció: un bit de màscara que ens permet habilitar o inhibir les peticions d'una línia i opcionalment un bit d'interrupció que ens indica si s'ha acceptat la petició, és a dir, si hi ha hagut una petició del mòdul d'E/S i estan habilitades les peticions d'interrupcions per a aquella línia.

Aquest sistema d'emmascarament també té 1 bit per a l'emmascarament general d'interrupcions, un bit de màscara que ens permet habilitar o inhibir totes les peticions d'interrupció emmascarables. Per a acceptar una petició d'una línia han de ser actius els bits de la línia i també el bit general d'interrupcions.

Si el processador en la fase de comprovació d'interrupció detecta que hi ha alguna petició d'interrupció, fa un cicle de reconeixement d'interrupció i es determina quina petició s'ha d'atendre accedint als bits de màscara del registre d'interrupcions. Si s'accepta una petició s'han d'inhibir les interrupcions d'aquella línia i de les línies menys prioritàries modificant els bits de màscara corresponents. En computadors simples o microcontroladors, el registre d'interrupció és visible al programador i hem d'identificar qui demana atenció i decidir qui atenem primer mitjançant un programa; en sistemes més complexos generalment es disposa d'un maquinari específic.

Aquesta forma d'emmascarament se sol utilitzar en sistemes amb poques línies de petició d'interrupció perquè, tot i ser molt flexible, si no es disposa d'un maquinari específic la gestió de prioritats s'ha de fer per programa i això la fa més lenta. És una situació semblant a l'enquesta que fem quan tenim una única línia de petició d'interrupció, però en lloc d'haver d'accedir als registres dels mòduls d'E/S mitjançant el bus del sistema, accedim a un registre del processador que és molt més fàcil i ràpid.

En l'**emmascarament per nivell**, a cada línia de petició d'interrupció s'hi assigna un **nivell d'interrupció**, anomenat també **nivell d'execució**. Si el nombre de nivells és inferior al nombre de línies de petició d'interrupció s'agrupen diverses línies a un mateix nivell.

Aquest nivell d'execució s'utilitza per a gestionar les prioritats, de manera que si ja atenem una petició d'interrupció només acceptem una nova petició i n'aturem l'atenció si aquesta petició té un nivell d'execució més prioritari.

El nivell d'execució es pot obtenir fàcilment connectant les línies de petició d'interrupció a un codificador de prioritat i el codi generat s'emmagatzema en un registre del processador, generalment en el registre d'estat del processador. Per a decidir si acceptem una determinada petició, amb un circuit comparador, hem de comparar el valor obtingut del codificador de prioritat amb el

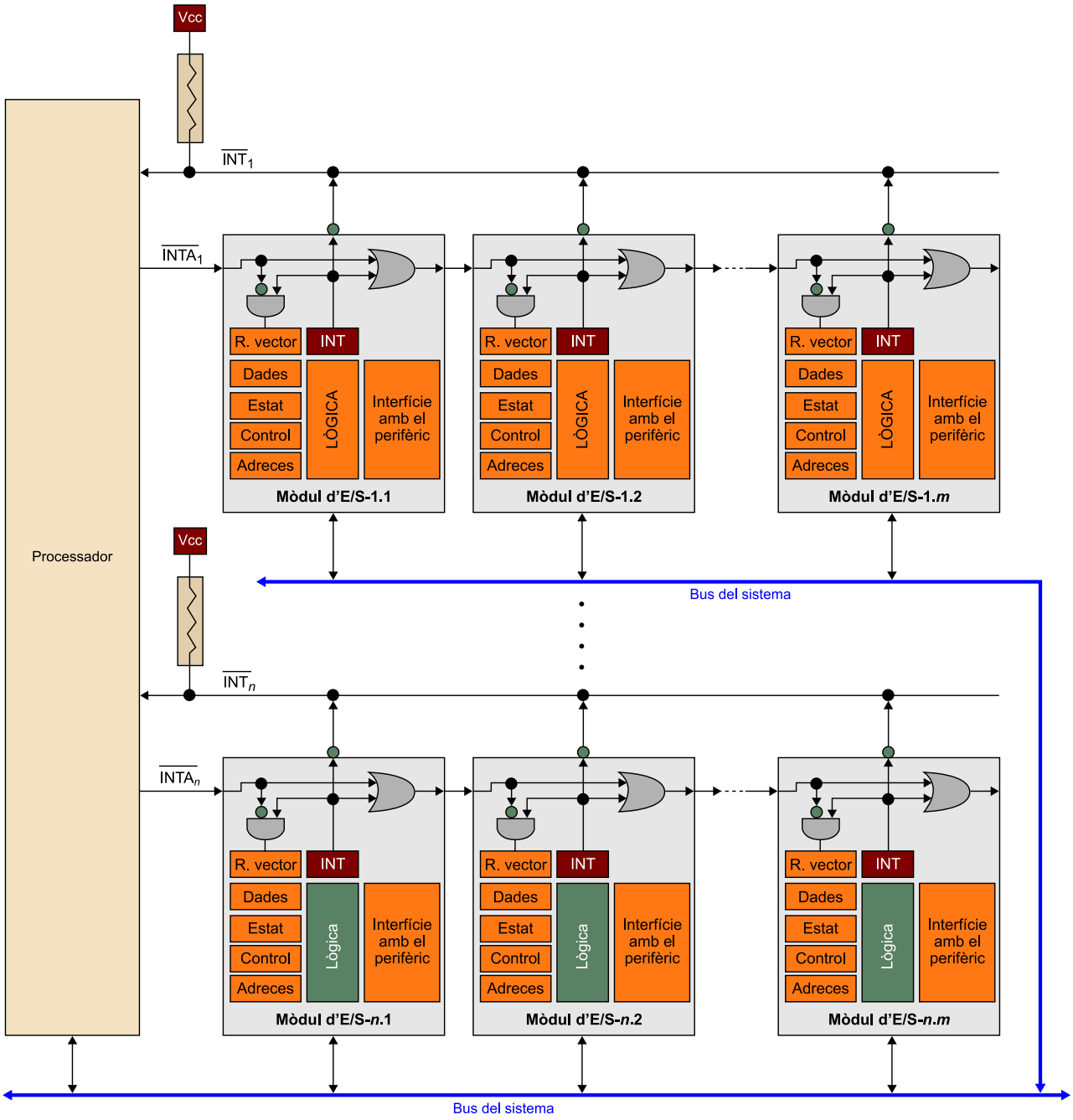
nivell d'execució que tenim emmagatzemat. Si és més prioritari el valor obtingut del codificador acceptem la petició i utilitzem aquest valor com un vector d'interrupció per a obtenir l'adreça de l'RSI per a atendre aquesta petició. El nou nivell d'execució s'actualitza de manera automàtica un cop feta la salvaguarda de l'estat del processador (s'ha emmagatzemat el PC i el registre d'estat a la pila del sistema) i comença l'execució de l'RSI. D'aquesta manera, quan acabem l'execució de l'RSI que atén aquesta petició més prioritària i restaurem l'estat del processador per reprendre l'execució del programa aturat, recuperem també el nivell d'execució que té ja que queda guardat amb la resta d'informació del registre d'estat.

Aquesta forma d'emmascarament és la més habitual en aquest tipus de sistemes.

Per altra banda, per a garantir un funcionament correcte del computador, el processador ha de disposar de línies de petició d'interrupció que no es puguin emmascarar. Només estan inhibides en moments concrets durant el cicle de reconeixement de les interrupcions i durant el retorn d'interrupció per a garantir l'estabilitat del sistema.

Els principals avantatges d'aquest sistema amb línies independents de petició d'interrupcions i de reconeixement són que permet la nidificació, la identificació del perifèric és molt ràpida i la gestió de prioritats molt flexible.

El desavantatge principal és que no es pot augmentar el nombre de mòduls d'E/S que podem atendre ja que implica un redisseny del processador o, com es pot veure en l'esquema següent, utilitzar sistemes híbrids establint per exemple un sistema d'encadenament (*daisy-chain*) en cadascuna de les línies de petició i reconeixement d'interrupció, tot i que aquesta opció fa la gestió de les interrupcions més complexa i lenta.



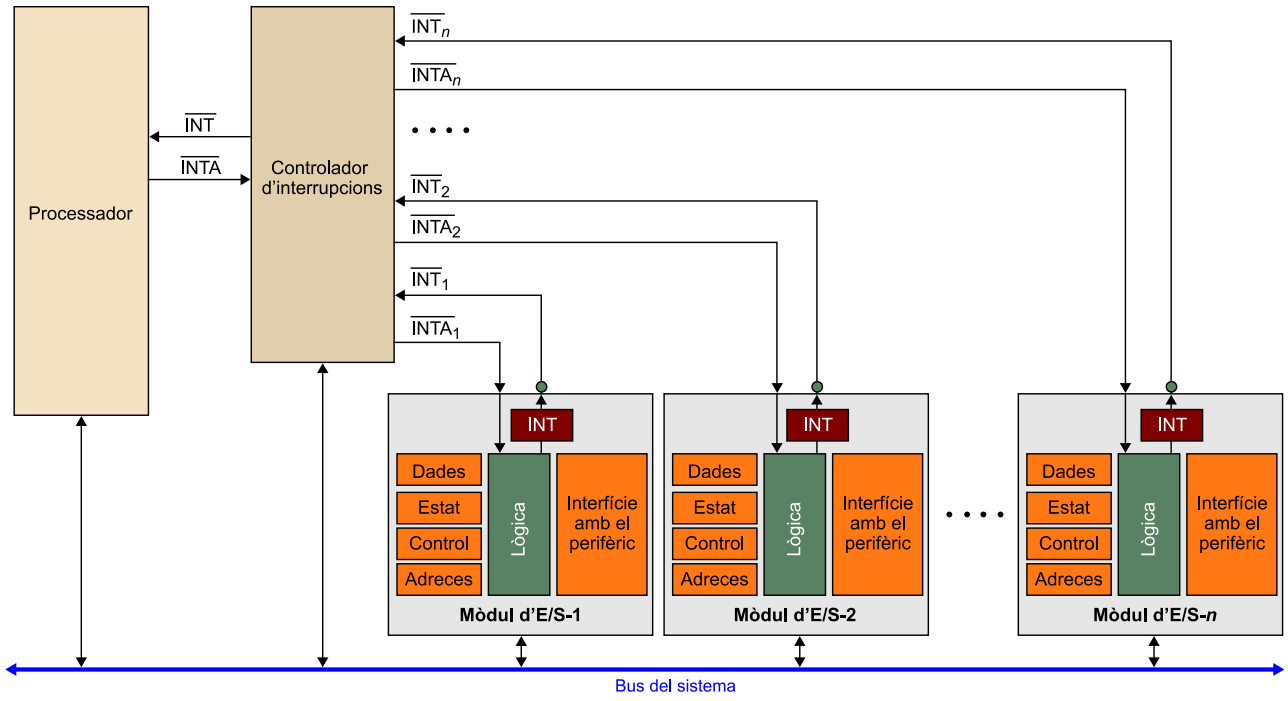
3.6. Sistema amb controladors d'interrupcions

En un sistema amb controladors d'interrupcions s'afegeix un nou element especialitzat per a gestionar les peticions d'interrupció que anomenem **controlador d'interrupcions**. Les línies de petició d'interrupció i de reconeixement de cada mòdul d'E/S estan connectades al controlador d'interrupcions. D'aquesta manera el processador només és necessari que tingui una línia de petició d'interrupció i una de reconeixement connectades al controlador d'interrupció per a gestionar les peticions provinents dels mòduls d'E/S.

Les funcions del controlador d'interrupció són les següents:

- Definir una política de prioritats per als mòduls d'E/S connectats al controlador.
- Identificar quin mòdul d'E/S demana atenció i informar-ne el processador.

La gestió d'una interrupció en aquest sistema és anàloga a la gestió d'una interrupció amb un únic mòdul d'E/S, i varia la fase de reconeixement de la interrupció per a gestionar les prioritats i identificar quin perifèric demana atenció.



El procés per a fer el reconeixement de la interrupció és el següent: per a identificar el perifèric que demana atenció i obtenir l'adreça de l'RSI que n'ha d'atendre la petició, utilitzem un sistema de vectorització molt semblant al descrit en el *daisy-chain*, però ara no fa falta que el mòdul d'E/S tingui un registre vector. El controlador d'interrupcions disposa d'un conjunt de registres vector on es guarden els vectors d'interrupció associats a cada línia.

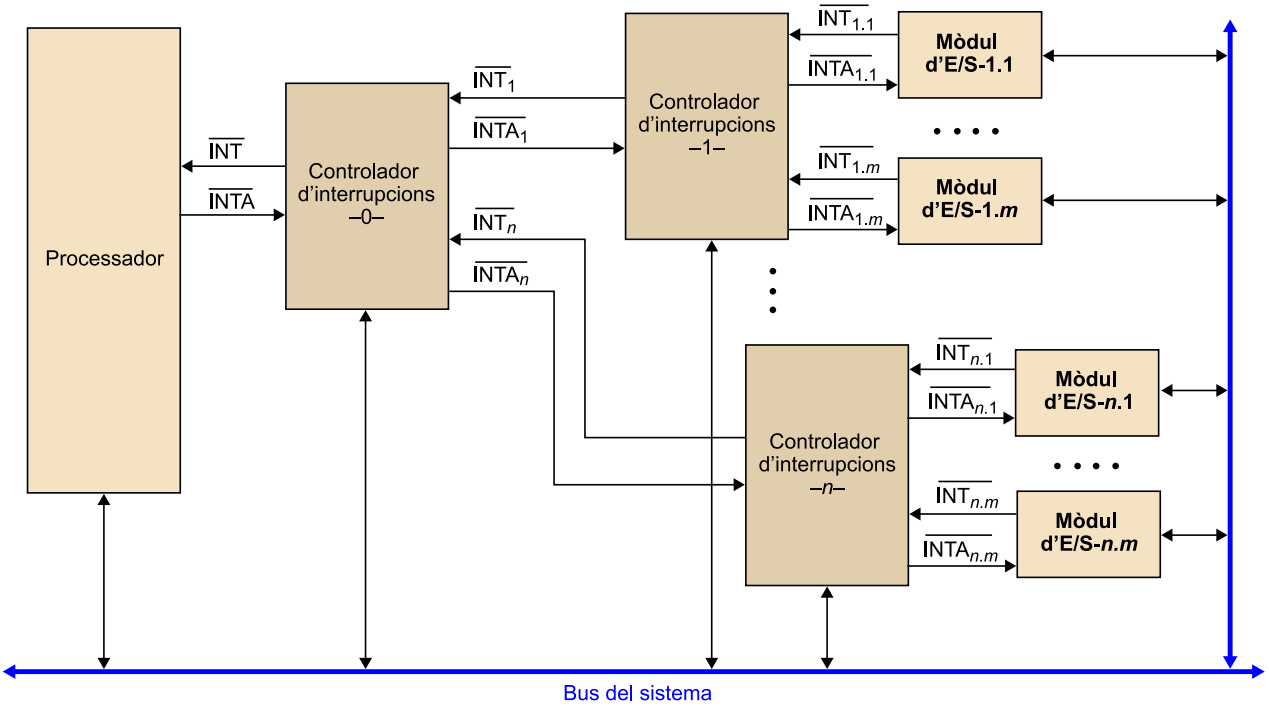
Quan arribi una petició d'interrupció d'un mòdul d'E/S prou prioritari, el controlador d'interrupcions fa la petició al processador activant el senyal INT i el processador contesta activant el senyal INTA. Quan el controlador d'interrupcions rep el senyal INTA col·loca el vector d'interrupció, corresponent al mòdul d'E/S que ha fet la petició, en el bus de dades i el processador llegeix el vector. D'aquesta manera el processador pot obtenir l'adreça de l'RSI

i començar l'execució de la rutina per fer la transferència de dades, comunicant-se directament amb el mòdul d'E/S. La transferència de dades no es gestiona mitjançant el controlador d'interrupcions.

Els principals avantatges d'aquest sistema amb un controlador d'interrupció són que la identificació del perifèric és força ràpida i la gestió de prioritats flexible, i també que el processador només necessita disposar d'una línia INT i una INTA per a gestionar múltiples mòduls d'E/S.

El desavantatge principal és que a causa de la gestió de prioritats que fa el controlador no es pot fer la salvaguarda del nivell d'execució de manera automàtica, sinó que s'ha de fer executant un petit programa que accedeixi al controlador d'interrupcions i fa el procés de salvaguarda de l'estat més lent.

Si tenim un sistema que utilitza controladors d'interrupció i volem augmentar el nombre de mòduls d'E/S hem de connectar el controlador d'interrupció en cascada com es mostra a continuació, però cal que els controladors estiguin dissenyats específicament per a connectar-los d'aquesta manera.



4. E/S amb accés directe a memòria

Les tècniques d'E/S que hem vist fins ara requereixen una dedicació important del processador (executar un fragment de codi) per a fer simples transferències de dades. Si volem transferir blocs de dades, aquestes tècniques encara posen més en evidència la ineficiència que tenen. En E/S programada implica que el processador no pugui fer res més i en E/S per interrupcions descarreguem el processador de la sincronització a costa de fer les rutines d'atenció més llargues per garantir l'estat del processador limitant-ne la velocitat de transferència.

En aquest apartat descriurem una tècnica molt més eficient per a transferir blocs de dades, l'**accés directe a memòria** (**DMA**). En aquesta tècnica el processador programa la transferència d'un bloc de dades entre el perifèric i la memòria encarregant a un nou element connectat al bus del sistema fer tota la transferència. Un cop acabada, aquest nou element avisa el processador. D'aquesta manera el processador pot dedicar tot el temps que dura la transferència del bloc a altres tasques. Aquest nou element que gestiona tota la transferència de dades entre el perifèric i la memòria principal l'anomenem **mòdul** o **controlador de DMA** o també en versions més evolucionades **canal** o **processador d'E/S**.

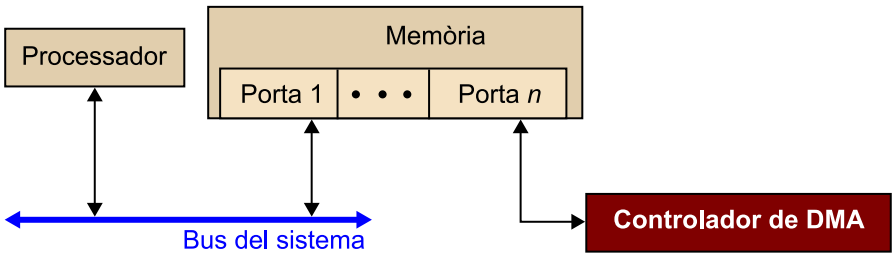
Utilitzant la tècnica d'E/S per DMA es descarrega el processador de la responsabilitat de dur a terme la sincronització i l'intercanvi de dades entre el perifèric i la memòria.

Per altra banda, ens apareix una nova problemàtica en el computador ja que hi ha dos dispositius, el processador i el controlador de DMA, que han d'accedir de manera concurrent a la memòria i cal establir un mecanisme per a resoldre aquest conflicte.

4.1. Accés concurrent a memòria

Bàsicament hi ha dues maneres de resoldre l'accés concurrent a memòria:

1) Connexions independents, utilitzant memòries multiporta. Hi ha una connexió independent per cada dispositiu que ha d'accedir a la memòria. Això permet al controlador de DMA accedir a la memòria sense que el processador hagi d'intervenir. Malgrat que l'afectació al processador és molt baixa, aquesta solució no se sol utilitzar perquè fa augmentar el cost de la memòria. També augmenta el temps d'accés, sobretot quan hi ha més d'una porta que vol accedir al mateix bloc d'informació.



2) Connexió compartida, utilitzant robatori de cicle. En aquest cas, la memòria només necessita una única porta i tant el processador com el controlador de DMA comparteixen el bus del sistema per a accedir a la memòria. En la majoria de sistemes, el processador és qui controla el bus; per tant, cal establir un mecanisme perquè el processador pugui cedir el bus al controlador de DMA i aquest controlador pugui fer l'intercanvi de les dades amb la memòria. Aquest mecanisme s'anomena *robatori de cicle* i és el que s'utilitza més freqüentment per a gestionar l'accés concurrent a la memòria.

Per a controlar l'accés al bus són necessaris dos senyals, BUSREQ i BUSACK (semblants als senyals INT i INTA utilitzats en interrupcions). Amb el senyal BUSREQ el controlador de DMA sol·licita el control del bus i el processador cedeix el bus activant el senyal BUSACK.

La diferència més important entre l'accés directe a memòria i la gestió d'interrupcions és que el processador pot inhibir les interrupcions totalment o parcialment, mentre que la cessió del bus no la pot inhibir i està obligat a cedir sempre el control del bus quan el controlador de DMA el sol·licita.

Els principals avantatges d'aquest sistema són que l'atenció és molt ràpida perquè no es pot inhibir la cessió del bus i que no és necessari guardar l'estat del processador, que només queda aturat durant un temps molt breu mentre el controlador de DMA té el control del bus.

La cessió del bus no és immediata. El processador només pot cedir el bus en acabar cadascuna de les fases del cicle d'execució de les instruccions. Un cop el controlador de DMA allibera el bus, de manera que s'acaba el robatori de cicle, el processador continua l'execució de la següent fase de la instrucció en curs. Per tant, el robatori de cicle es pot produir en diferents punts dins del cicle d'execució d'una instrucció.

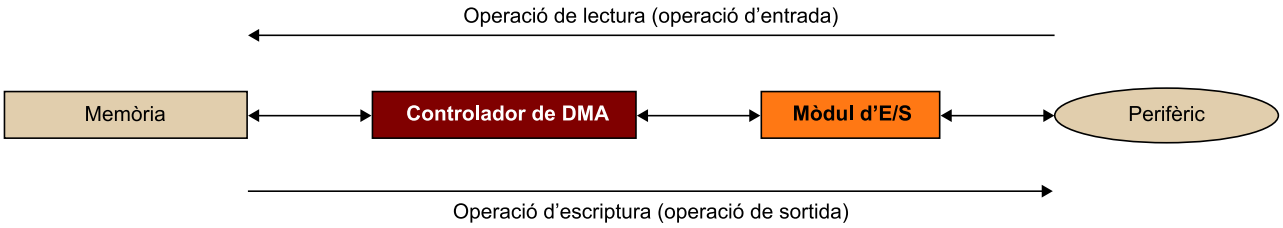
El principal desavantatge d'aquest sistema és que s'allarga el temps d'execució de les instruccions a causa dels possibles robatoris de cicle que es poden produir.

4.2. Operació d'E/S amb accés directe a memòria

El procés per a fer una transferència d'E/S utilitzant aquesta tècnica és el següent:

1) Programació de l'operació d'E/S: el processador envia la informació necessària al controlador de DMA perquè aquest controlador pugui gestionar tota la transferència de dades. Un cop acaba la programació de l'operació d'E/S el processador pot executar altres programes mentre es fa la transferència.

2) Transferència del bloc de dades: les dues operacions bàsiques que fa el controlador de DMA són la lectura d'un bloc de dades d'un perifèric i l'escriptura d'un bloc de dades en un perifèric, encara que també pot fer altres operacions. Totes les dades de la transferència passen pel controlador de DMA.



3) Finalització de l'operació d'E/S: quan s'ha acabat la transferència del bloc el controlador de DMA envia una petició d'interrupció al processador per informar que s'ha acabat la transferència de dades.

Nota

Per a utilitzar aquesta tècnica d'E/S en un computador, és necessari considerar tant aspectes del programari com del maquinari.

4.3. Controladors de DMA

En un sistema d'aquest tipus les implicacions pel que fa al maquinari i al sistema de connexió d'aquest maquinari són diverses.

El processador ha de disposar d'un sistema per a gestionar interrupcions, com hem explicat en el apartat anterior. El controlador de DMA avisa el processador que s'ha acabat la transferència del bloc de dades mitjançant una petició d'interrupció.

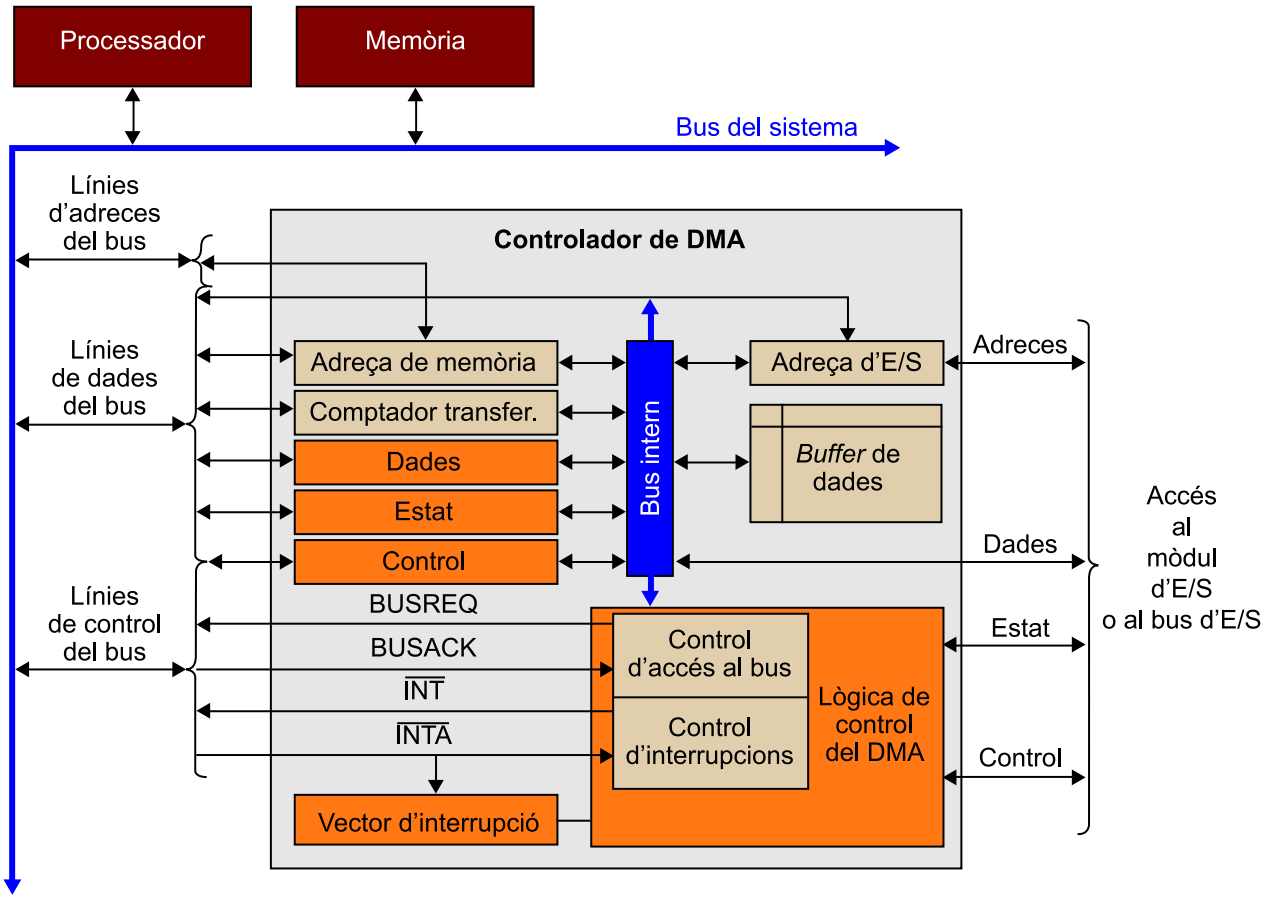
Els elements bàsics que ha de tenir el controlador de DMA per a gestionar una transferència de dades entre el perifèric i la memòria són els següents:

- Un banc de registres per a gestionar les operacions d'E/S.
- La lògica de control necessària per a gestionar les transferències entre la memòria i el mòdul d'E/S.
- La lògica necessària per a gestionar la interrupció de la finalització de l'operació d'E/S.

- Senyals de control per a accedir al bus del sistema i per a la gestió d'interrupcions (BUSREQ, BUSACK, INT, INTA...).

El banc de registres del controlador de DMA està format pels registres següents:

- **Registre de control**: s'utilitza generalment per a donar les ordres al controlador.
- **Registre d'estat**: dóna informació de l'estat de la transferència de dades.
- **Registre de dades**: emmagatzema les dades que es vol intercanviar.
- **Registre d'adreces de memòria**: indica l'adreça de memòria on es llegiran o s'escriuran les dades que s'han de transferir a cada moment.
- **Registre comptador**: indica inicialment el nombre de transferències que s'ha de fer i s'anirà actualitzant durant la transferència fins que valgui zero i indicarà a cada moment el nombre de transferències que queden per fer.
- **Registre d'adreces d'E/S**: indica la posició dins del perifèric on es llegiran o s'escriuran les dades que s'han de transferir a cada moment.

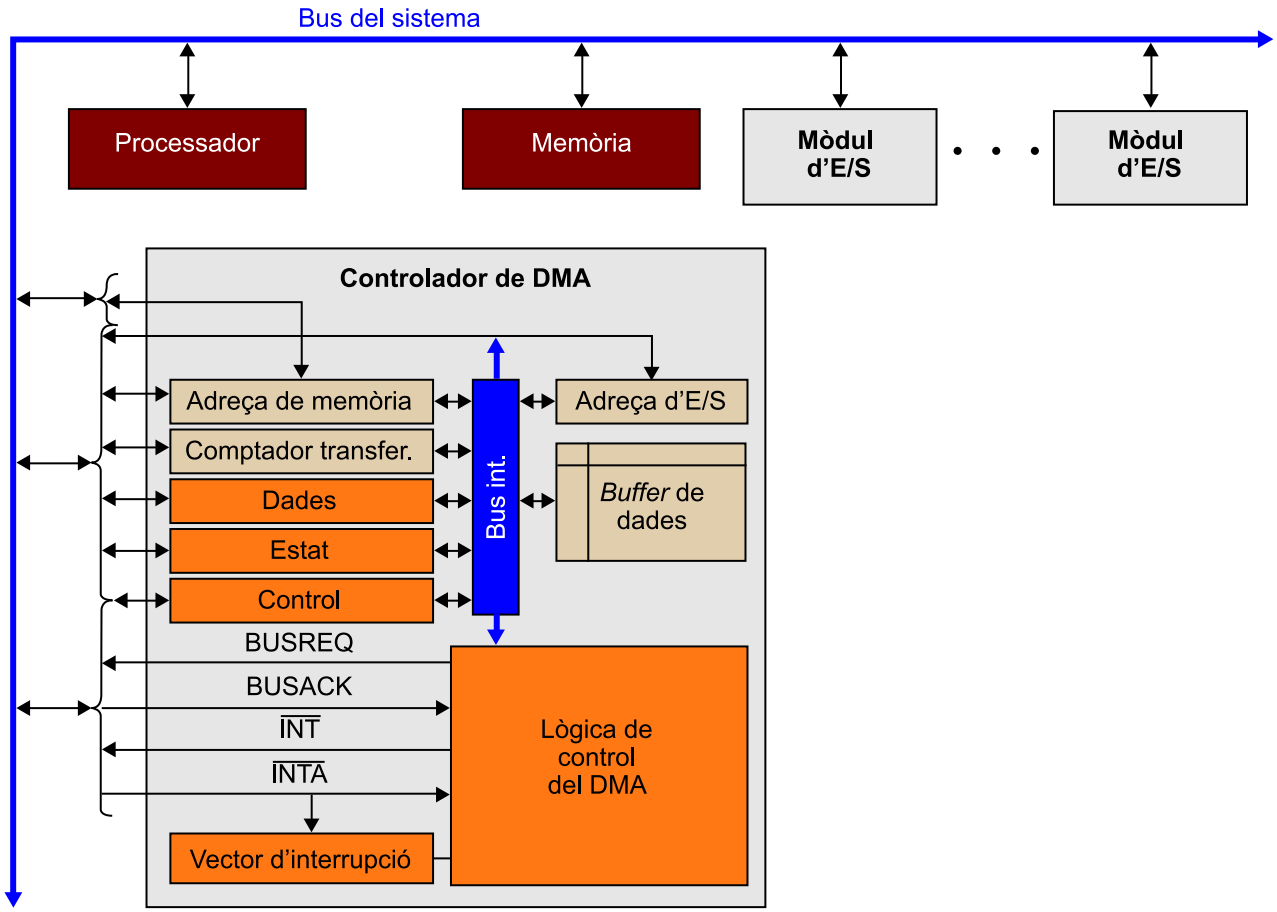


4.3.1. Formes de connexió dels controladors de DMA

A continuació tractarem les configuracions més habituals del controlador de DMA.

La configuració següent és la més simple però també la més ineficient perquè s'ha de fer dos accessos al bus: un per a accedir a memòria i un altre per a accedir al mòdul d'E/S.

Connexió del controlador de DMA i dels mòduls d'E/S amb un únic bus



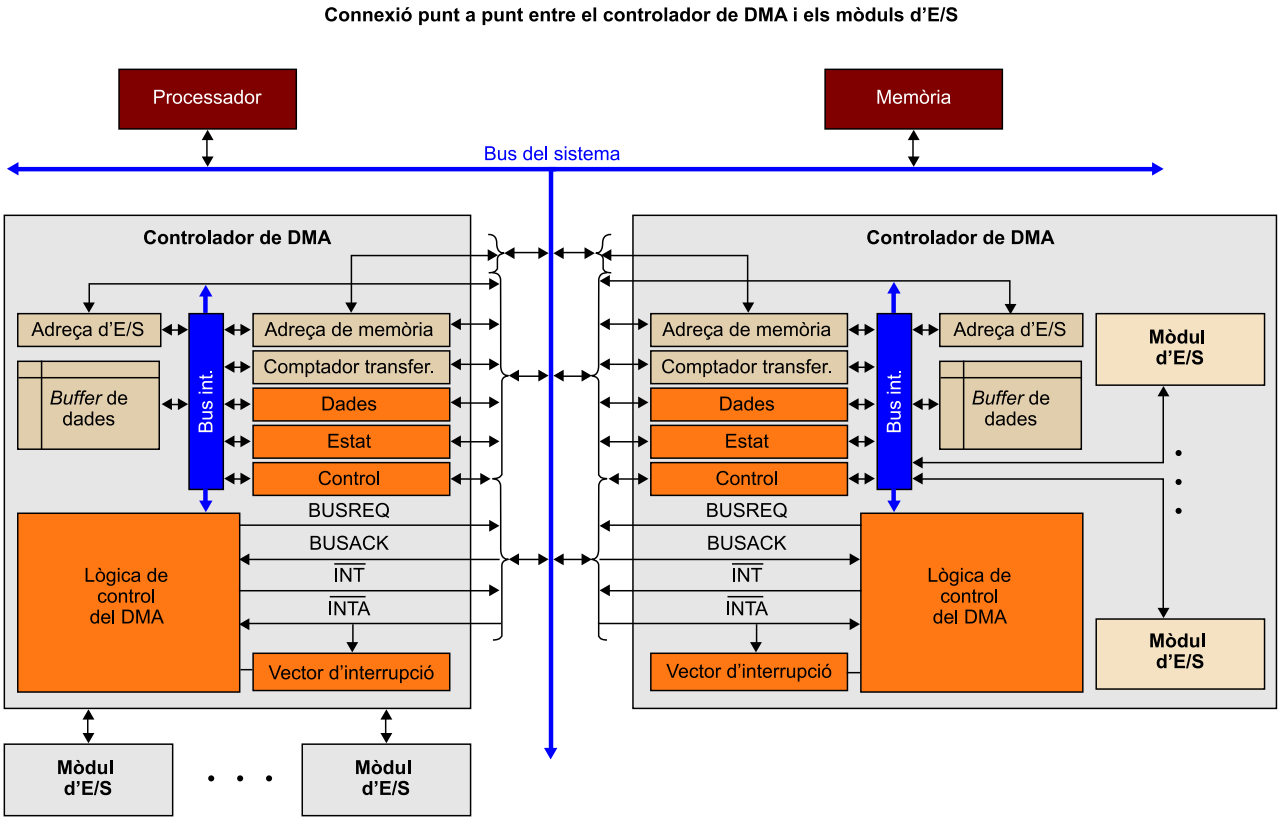
En cas de tenir més d'un controlador de DMA connectat al bus la gestió de prioritats per a decidir a qui se cedeix el bus es fa de manera molt semblant a la gestió de prioritats explicada en l'E/S per interrupcions:

- Sistema amb encadenament (*daisy-chain*), en què es fa l'encadenament del senyal BUSACK.

- Utilitzar un controlador de DMA que gestioni múltiples transferències, de manera semblant al controlador d'interrupcions, però ara la transferència de dades amb la memòria es fa mitjançant el controlador de DMA i no directament entre el mòdul d'E/S i el processador com passa quan utilitzem controladors d'interrupcions. En aquest cas, tant podem trobar con-

nexions independents entre el controlador i cada mòdul d'E/S com tots els mòduls connectats a un bus específic d'E/S.

Una millora de la configuració anterior consisteix a connectar els mòduls d'E/S directament al controlador de DMA, de manera que necessitem només un accés al bus del sistema. Aquesta configuració també permet connectar-hi controladors de DMA que integren la lògica del mòdul d'E/S.

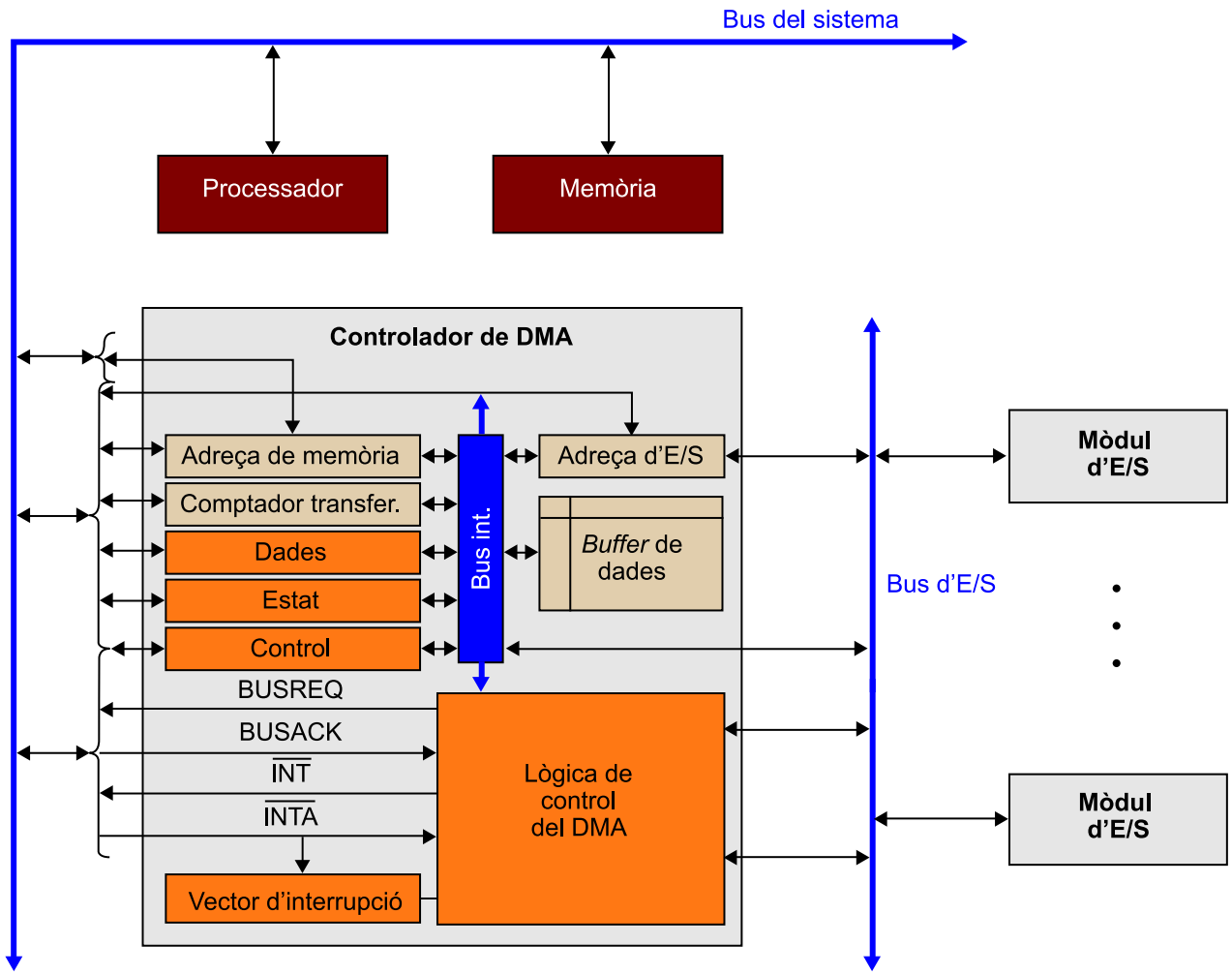


Transferències simultànies

Si s'ha de fer les operacions d'E/S amb diversos perifèrics simultàniament hem de replicar part del conjunt de registres (adreces, comptador, bits de control...) tants cops com transferències de dades simultànies vulguem gestionar.

Una altra configuració possible consisteix a connectar els mòduls d'E/S al controlador de DMA mitjançant un bus d'E/S. El flux d'informació entre el controlador de DMA i els mòduls d'E/S no interfereix amb els accessos al bus del sistema.

Connexió del controlador de DMA i dels mòduls d'E/S mitjançant un bus d'E/S



4.3.2. Operació d'E/S mitjançant un controlador de DMA

Per a fer una transferència d'E/S utilitzant aquesta tècnica se segueixen els passos següents:

1) Programació de l'operació d'E/S. El processador ha d'enviar la informació necessària al controlador de DMA perquè aquest controlador pugui fer la transferència de manera autònoma. Aquesta informació s'ha d'escriure en el banc de registres del controlador de DMA.

La informació imprescindible que s'ha d'escriure és la següent:

a) Operació que cal fer. Les operacions bàsiques són de lectura o escriptura, però també es poden fer altres tipus d'operacions de control del perifèric. Aquesta informació s'escriu en el registre de control mitjançant les línies de control o les línies de dades del bus de sistema.

b) Adreça de memòria. Adreça inicial de memòria on es llegiran o s'escriuran les dades que s'han de transferir considerant que el bloc de dades que mourem és en adreces contigües de memòria. Aquesta informació s'escriu en el registre d'adreces de memòria.

c) Mida del bloc de dades. Nombre de transferències que s'ha de fer mitjançant el bus del sistema per a transferir tot el bloc de dades. Aquesta informació s'escriu en el registre comptador.

d) Adreça del perifèric. Posició inicial dins del perifèric on es llegiran o s'escriuran les dades que s'han de transferir. Aquesta informació s'escriu en el registre d'adreces d'E/S. Aquesta informació depèn del perifèric i en certa mesura del tipus d'informació que s'ha de transmetre. Si és un dispositiu d'emmagatzematge hem de saber on està guardat per a recuperar després les dades.

2) Transferència del bloc de dades. Atès que el processador ha delegat al controlador de DMA l'operació d'E/S, el controlador de DMA ha de fer la transferència de dades de tot el bloc. La transferència es fa dada a dada accedint directament a memòria sense la intervenció del processador.

Els passos per a una operació de lectura i per a una operació d'escriptura són els següents:

Després de transferir cada dada el controlador de DMA decrementa el registre comptador i actualitza el registre d'adreces de memòria amb l'adreça on hem de llegir o emmagatzemar la dada següent. Quan el registre comptador arriba a zero, el controlador de DMA genera una petició d'interrupció per avisar el processador que s'ha acabat la transferència del bloc.

3) Finalització de l'operació d'E/S. Un cop el controlador de DMA envia la petició d'interrupció al processador per informar que s'ha acabat la transferència del bloc de dades, el processador executa l'RSI corresponent per acabar l'operació d'E/S.

Cal notar que el controlador de DMA s'ha de dotar del maquinari necessari perquè pugui generar la petició d'interrupció segons el sistema d'interrupcions que utilitza el processador; per exemple, si fem servir un sistema amb encadenament (*daisy-chain*) hem de poder generar una INT, rebre una INTA i la lògica per propagar-la i un registre vector per identificar-se.

4.4. Controlador de DMA en mode ràfega

Una manera d'optimitzar les operacions d'E/S per DMA consisteix a reduir el nombre de cessions i recuperacions del bus. Per a fer-ho, en lloc de sol·licitar i alliberar el bus per a cada dada que s'ha de transferir, se sol·licita i s'allibera el bus per a transferir un conjunt de dades de manera consecutiva. Aquesta modalitat de transferència s'anomena **mode ràfega**.

Per a fer la transferència d'aquest conjunt de dades, que anomenem **ràfega**, el controlador de DMA cal que disposi d'una memòria intermèdia (*buffer*), de manera que la transferència de dades entre la memòria i el controlador de DMA és pugui fer a la velocitat que permeti la memòria i no quedant limitada a la velocitat del perifèric.

Aquest mode de funcionament no afecta la programació ni la finalització de l'operació d'E/S descrita anteriorment, però sí que modifica la transferència de dades.

El funcionament de la transferència del bloc de dades és el següent: en el cas de la lectura, cada cop que el mòdul d'E/S té una dada disponible el controlador de DMA l'emmagatzema en la memòria intermèdia i decrementa el registre comptador. Quan la memòria intermèdia és plena o el comptador ha arribat a zero, sol·licita el bus. Un cop el processador li cedeix el bus, escriu a memòria tot el conjunt de dades emmagatzemades en la memòria intermèdia, i fa tants accessos a memòria com dades tenim i actualitza el registre d'adreces de memòria en cada accés. En acabar la transferència del conjunt de dades allibera el bus.

En el cas de l'escriptura, el controlador de DMA sol·licita el bus i, quan el processador cedeix el bus, el controlador de DMA llegeix una dada de la memòria, l'emmagatzema en la memòria intermèdia, decrementa el registre comptador i actualitza el registre d'adreces de memòria. Quan la memòria intermèdia és plena o el comptador ha arribat a zero, allibera el bus. A continuació transfereix tot aquest conjunt de dades emmagatzemades en la memòria intermèdia al mòdul d'E/S, i espera per a cada dada que el mòdul d'E/S estigui preparat.

Un cop acabada una ràfega, si el registre comptador no ha arribat a zero, comença la transferència d'una nova ràfega.

4.5. Canals d'E/S

Els canals d'E/S són una millora dels controladors de DMA. Poden executar instruccions que llegeixen directament de memòria. Això permet gestionar amb més autonomia les operacions d'E/S i d'aquesta manera es pot controlar múltiples operacions d'E/S amb dispositius amb una mínima intervenció del processador.

Aquests canals encara es poden fer més complexos afegint-hi una memòria local pròpia que els converteix en processadors específics d'E/S.

La programació de l'operació d'E/S per part del processador es fa escrivint en memòria les dades i les instruccions que necessita el canal d'E/S per a gestionar tota l'operació d'E/S. La informació que s'especifica inclou el dispositiu a què hem d'accedir, l'operació que s'ha de fer indicant el nivell de prioritat, l'adreça del bloc de dades on hem de llegir o escriure les dades que s'han de transferir, la mida del bloc de dades que s'han de transferir, com s'ha de fer el tractament d'errors i com ha d'informar el processador del final de l'operació d'E/S.

Quan s'acaba l'operació d'E/S el canal d'E/S informa el processador que s'ha acabat la transferència i de possibles errors mitjançant la memòria. També es pot indicar el final de la transferència mitjançant interrupcions.

Les dues configuracions bàsiques de canals d'E/S són les següents:

- **Canal selector**: està dissenyat per a perifèrics d'alta velocitat de transferència i només permet una operació de transferència simultània.

- **Canal multiplexor**: està dissenyat per a perifèrics més lents de transferència i pot combinar la transferència de blocs de dades de diferents dispositius.

Els avantatges principals dels canals d'E/S respecte dels controladors d'E/S són els següents:

- Permeten controlar operacions d'E/S simultànies.

- Es pot programar múltiples operacions d'E/S sobre diferents dispositius o seqüències d'operacions sobre el mateix dispositiu, mentre el canal d'E/S fa altres operacions d'E/S.

5. Comparació de les tècniques d'E/S

Per a analitzar les prestacions d'un sistema d'E/S compararem les diferents tècniques d'E/S estudiades i així aprofundirem una mica més en l'estudi del funcionament d'aquestes tècniques, i també definirem els paràmetres bàsics d'una transferència d'E/S per determinar la dedicació del processador en cadascuna de les tècniques d'E/S.

Per a fer l'anàlisi de les prestacions considerem que fem la transferència d'un bloc de dades entre un perifèric i el computador.

$$N_{dades} = m_{bloc} / m_{dada}$$

On

- m_{dada}: mida d'una dada expressada en bytes. Entenem *dada* com la unitat bàsica de transferència.
- m_{bloc}: mida del bloc de dades que volem transferir entre el computador i el perifèric expressada en bytes.
- N_{dades}: nombre de dades que formen el bloc que volem transferir. Ens indica també el nombre de transferències que s'han de fer.

Perifèric

Primer cal analitzar com condiciona la transferència d'E/S el perifèric. Per a fer-ho definim els paràmetres següents:

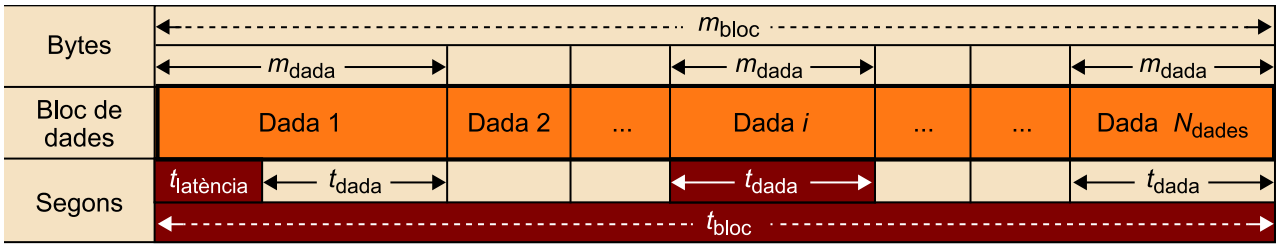
- v_{transf}: velocitat de transferència mitjana del perifèric expressada en bytes per segon.
- t_{dada}: temps de transferència d'una dada (entre el perifèric i el mòdul d'E/S) expressat en segons.

$$t_{dada} = m_{dada} / v_{transf}$$

- $t_{latència}$: temps de latència, temps que necessita el perifèric per a iniciar la primera transferència expressat en segons. En alguns perifèrics aquest temps pot ser significatiu.

- t_{bloc}: temps de transferència de les dades de tot el bloc (entre el perifèric i el mòdul d'E/S) expressat en segons.

$$t_{bloc} = t_{latència} + (N_{dades} \cdot t_{dada})$$



Definim a continuació els paràmetres bàsics que determinen d'alguna manera la transferència de dades en les diferents tècniques d'E/S.

Processador

- t_{inst}: temps mitjà d'execució d'una instrucció expressat en segons.

Bus del sistema i memòria

- $t_{cessió}$: temps necessari perquè el processador faci la cessió del bus expressat en segons.

- t_{recup}: temps necessari perquè el processador faci la recuperació del bus expressat en segons.

- t_{mem}: temps mitjà per a fer una lectura o una escriptura en la memòria principal mitjançant el bus del sistema expressat en segons.

Transferència d'E/S

Tal com hem definit anteriorment en una transferència d'E/S, la programació i la finalització de la transferència són passos comuns a totes les tècniques d'E/S i són responsabilitat del processador. Cal tenir present que en E/S per DMA la programació de la transferència és estrictament necessària i té un pes més important que en la resta de tècniques.

- t_{prog}: temps necessari per a fer la programació de la transferència d'E/S expressat en segons.

- t_{final}: temps necessari per a fer la finalització de la transferència d'E/S expressat en segons.

Aquests temps es poden calcular a partir del nombre d'instruccions que cal executar per a fer la programació (N_{prog}) o la finalització de la transferència (N_{final}), multiplicant el nombre d'instruccions necessàries pel temps mitjà d'execució d'una instrucció (t_{inst}).

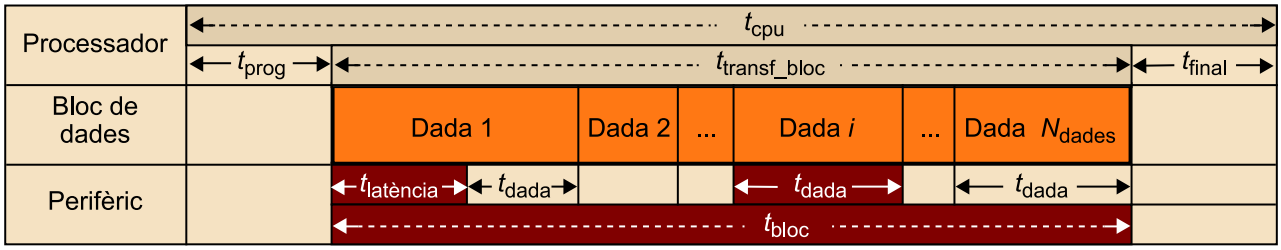
- t_{transf_dada}: temps que el processador està ocupat durant la transferència d'una dada amb el mòdul d'E/S o aturat mentre el controlador de DMA fa la transferència d'una dada expressat en segons.

- t_{transf_bloc}: temps que el processador està ocupat durant la transferència de dades de tot el bloc amb el mòdul d'E/S o aturat mentre el DMA fa la transferència de dades de tot el bloc expressat en segons.

t_{transf_dada} i t_{transf_bloc} dependran de la tècnica d'E/S utilitzada per a fer la transferència.

- t_{cpu}: temps que el processador està ocupat o aturat durant la transferència d'E/S expressat en segons.

$$t_{cpu} = t_{prog} + t_{transf\_bloc} + t_{final}$$

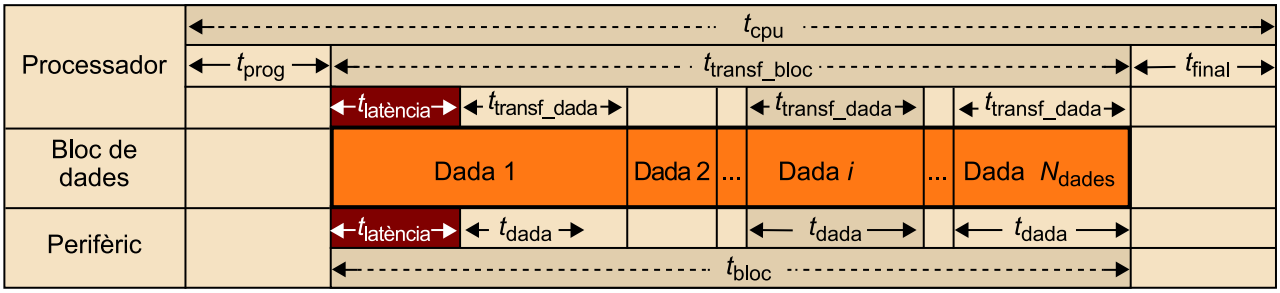


Fins ara hem analitzat els paràmetres bàsics que condicionen els passos principals en una transferència d'E/S. A continuació, veurem com afecten les diferents tècniques d'E/S el temps que el processador està ocupat o aturat (t_{transf_bloc} i t_{cpu}).

E/S programada

Quan utilitzem aquesta tècnica el processador està dedicat el cent per cent del temps a aquesta tasca; per tant, no té temps per a executar altres instruccions:

$$t_{dada} = m_{dada} / v_{transf}$$
$$t_{transf\_dada} = t_{dada}$$
$$t_{transf\_bloc} = t_{latència} + (N_{dades} \cdot t_{transf\_dada}) = t_{bloc}$$
$$t_{cpu} = t_{prog} + t_{transf\_bloc} + t_{final}$$



Cal notar que aquests temps no depenen del nombre d'instruccions del bucle de sincronització, ni del codi per a fer la transferència de la dada, sinó que només depenen de la velocitat de transferència del perifèric.

E/S per interrupcions

Quan utilitzem E/S per interrupcions, la sincronització és responsabilitat del mòdul d'E/S i el processador només és responsable de l'operació de transferència de les dades. La transferència es fa quan arriba la petició d'interrupció (INT) del mòdul d'E/S i el processador executa l'RSI per atendre-la.

Per a determinar el temps que dedica el processador a atendre la transferència d'una dada definim els paràmetres següents:

- t_{rec_int}: temps que passa des que el processador reconeix que hi ha una petició d'interrupció fins que es pot iniciar l'execució de l'RSI expressat en segons. Aquest temps inclou el reconeixement de la interrupció i la salvaguarda de l'estat del processador.

- t_{rsi}: temps d'execució de l'RSI expressat en segons. Aquest temps inclou inhibir les interrupcions si és necessari, guardar a la pila els registres que es poden modificar, accedir al mòdul d'E/S per a fer la transferència de la dada, restaurar de la pila els registres, tornar a habilitar les interrupcions i fer el retorn d'interrupció.

- N_{rsi}: nombre d'instruccions de l'RSI.

Tant el temps de reconeixement de la interrupció i salvaguarda de l'estat (t_{rec_int}) com el temps d'execució de l'RSI (t_{rsi}) depenen de diferents factors que cal tenir en compte si s'han de calcular.

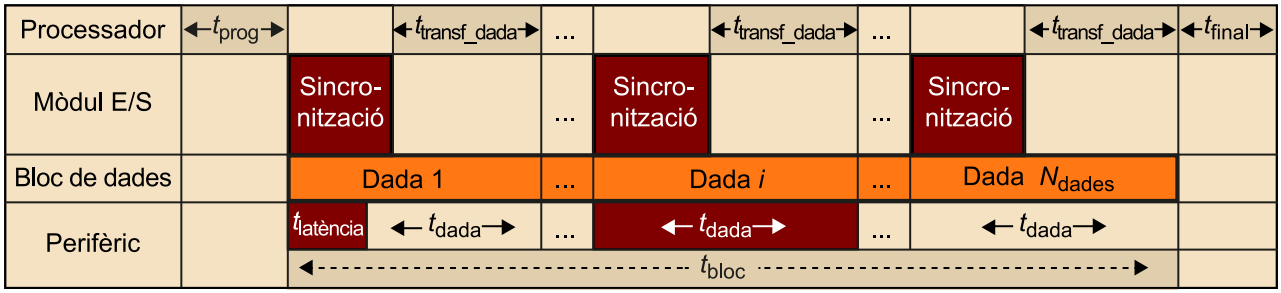
De manera general diem que:

$$t_{rsi} = N_{rsi} \cdot t_{inst}$$

El temps total que el processador està ocupat durant tota la transferència d'E/S es pot calcular de la manera següent:

$$t_{transf\_dada} = t_{rec\_int} + t_{rsi}$$
$$t_{transf\_bloc} = N_{dades} \cdot t_{transf\_dada}$$
$$t_{cpu} = t_{prog} + t_{transf\_bloc} + t_{final}$$

Com que en aquesta tècnica d'E/S la fase de sincronització és responsabilitat del mòdul d'E/S, durant el temps que dura la sincronització el processador podrà executar altres programes i de forma general podem dir que t_{transf_dada} és més petit que t_{dada}.



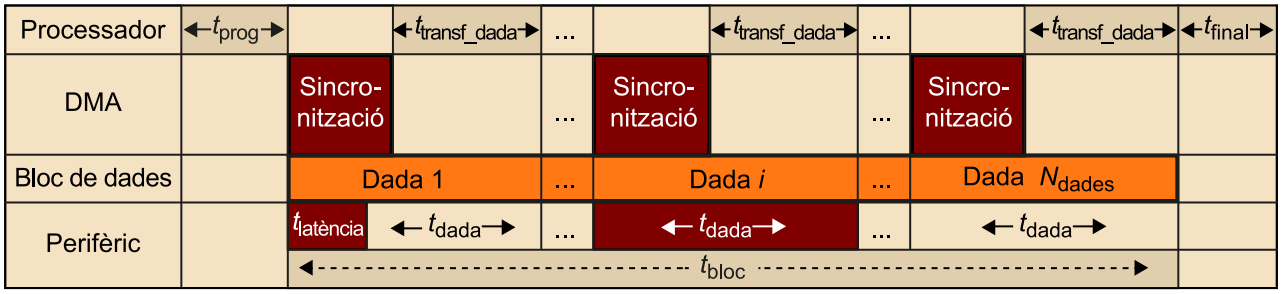
E/S per DMA

En aquesta tècnica d'E/S es descarrega el processador de la responsabilitat de dur a terme la transferència de les dades. Aquesta responsabilitat es deixa al controlador de DMA. Aquest controlador fa la transferència de les dades, però per a fer-ho ha d'utilitzar el bus del sistema amb la tècnica del robatori de cicle.

Vegem quant temps està ocupat o aturat el processador mentre dura la transferència de les dades. El processador està ocupat durant la programació i la finalització de la transferència i està aturat mentre el controlador de DMA utilitza el bus del sistema per a fer la transferència de les dades. Per a fer-ho utilitzem els paràmetres següents:

$$t_{transf\_dada} = t_{cessió} + t_{mem} + t_{recup}$$
$$t_{transf\_bloc} = N_{dades} \cdot t_{transf\_dada}$$
$$t_{cpu} = t_{prog} + t_{transf\_bloc} + t_{final}$$

Com que en aquesta tècnica d'E/S la fase de sincronització és responsabilitat del DMA, durant el temps que dura la sincronització el processador podrà executar altres programes i de forma general podem dir que t_{transf_dada} és més petit que t_{dada}.



E/S per DMA en mode ràfega

En aquest cas, es considera que la unitat de transferència del DMA està formada per un conjunt de dades. Cada cop que se sol·licita el bus, en lloc de transferir una única dada, es transfereix un conjunt de dades. Cada transferència d'un conjunt de dades l'anomenem *ràfega*. Això es pot fer si el controlador de DMA disposa d'una memòria intermèdia (*buffer*) per a emmagatzemar el conjunt de dades que transferim en cada ràfega.

- $N_{ràfega}$: nombre de dades que formen una ràfega.

Si sabem la mida de la memòria intermèdia, podem obtenir el nombre de dades de la ràfega de la manera següent:

- m_{buffer}: mida de la memòria intermèdia de dades expressada en bytes.

$$N_{ràfega} = m_{buffer} / m_{dada}$$

El temps que el processador està ocupat o aturat durant la transferència es pot calcular de la manera següent:

$$t_{transf\_ràfega} = t_{cessió} + (N_{ràfega} \cdot t_{mem}) + t_{recup}$$
$$t_{transf\_bloc} = (N_{dades} / N_{ràfega}) \cdot t_{transf\_ràfega}$$
$$t_{cpu} = t_{prog} + t_{transf\_bloc} + t_{final}$$

Ocupació del processador

Vegem quina és la proporció de temps que el processador dedica a la transferència d'E/S respecte del temps que triga el perifèric a fer tota la transferència. El percentatge d'ocupació del processador.

Com que el perifèric, llevat de casos excepcionals, és més lent que el processador, és el perifèric qui determina el temps necessari per a fer la transferència d'un bloc de dades.

$$t_{transf\_bloc} \leq t_{bloc}$$

Considerant tota la transferència d'E/S:

$$\%_{ocupació} = (t_{transf\_bloc} \cdot 100) / t_{bloc}$$

- t_{disp}: temps que el processador és lliure per a fer altres tasques durant la transferència d'E/S.

$$t_{disp} = (t_{bloc} - t_{transf\_bloc})$$

Exemple

Fem una comparació de les tres tècniques d'E/S utilitzant dades concretes per a transferir un bloc de dades.

Definim primer els paràmetres bàsics d'aquesta transferència:

m_{dada}	mida de la dada (bus de dades i registres)	4 Bytes
m_{bloc}	mida del bloc que es vol transferir	8 MBytes

Processador:

	freqüència del rellotge del processador	2 GHz
	nombre d'instruccions per cicle de rellotge	4
t_{cicle}	temps d'un cicle de rellotge (1 / 2 GHz)	0,5 ns
t_{inst}	temps d'execució de les instruccions (t_{cicle} / 4)	0,125 ns

Bus del sistema i memòria:

$t_{cessió}$	temps per a cedir el bus	0,5 ns
t_{recup}	temps per a alliberar el bus	1 ns
t_{mem}	temps de lectura/escriptura a memòria	0,5 ns

Recordeu-ho

Submúltiples de la unitat de temps (segons):

10^{-3}: mil·li (m)

10^{-6}: micro (μ)

10^{-9}: nano (n)

10^{-12}: pico (p)

Múltiples de les unitats de dades (bits o bytes).

2^{10}: kilo (K)

2^{20}: mega (M)

2^{30}: giga (G)

2^{40}: tera (T)

Freqüència

La freqüència ens indica el nombre de cicles de rellotge per segon. Calculant la inversa de la freqüència obtenim el temps d'un cicle de rellotge del processador.

t_{cicle} = 1/freqüència

Perifèric:

v_{transf}	velocitat de transferència	128 MBytes/s
$t_{latència}$	latència	7 ms

Calculem el nombre de dades que formen el bloc que volem transferir, que ens indica també el nombre de transferències que s'han de fer, el temps de transferència d'una dada i a partir d'aquest el temps de transferir un bloc.

N_{dades}	m_{bloc} / m_{dada}	$8 \cdot 2^{20}$ / 4 = $2 \cdot 2^{20}$
t_{dada}	m_{dada} / v_{transf}	(4 / $128 \cdot 2^{20}$)s = 0,000000029802 s = 29,802 ns
t_{bloc}	$t_{latència}$ + (N_{dades} · t_{dada})	0,007 + ($2 \cdot 2^{20}$ · 0,000000029802) = 0,007 + 0,0625 = 0,0695 s = 69,5 ms

Tècniques d'E/S

Ara analitzem els temps de transferència per a cadascuna de les tècniques d'E/S.

E/S programada:

N_{prog}	programar la transferència	15 instruccions
N_{final}	finalitzar la transferència	10 instruccions

t_{prog}	N_{prog} · t_{inst}	15 · 0,125 = 1,875 ns
t_{final}	N_{final} · t_{inst}	10 · 0,125 = 1,25 ns
t_{transf_dada}	t_{dada}	29,802 ns
t_{transf_bloc}	t_{bloc}	69,5 ms
t_{cpu}	t_{prog} + t_{transf_bloc} + t_{final}	1,875 + 69.500.000 + 1,25 = 69.500.003 ns

E/S per interrupcions:

N_{prog}	programar la transferència	20 instruccions
N_{final}	finalitzar la transferència	10 instruccions
t_{rec_int}	temps de reconeixement de la interrupció	8 ns
N_{rsi}	nombre d'instruccions de l'RSI	12 instruccions

t_{prog}	N_{prog} · t_{inst}	20 · 0,125 = 2,5 ns
t_{final}	N_{final} · t_{inst}	10 · 0,125 = 1,25 ns
t_{rsi}	N_{rsi} · t_{inst}	12 · 0,125 = 1,5 ns
t_{transf_dada}	t_{rec_int} + t_{rsi}	8 + 1,5 = 9,5 ns

t_{transf_bloc}	$N_{dades} \cdot t_{transf\_dada}$	$2 \cdot 2^{20} \cdot 9,5 = 19,92$ ms = 19.922.944 ns
t_{cpu}	$t_{prog} + t_{transf\_bloc} + t_{final}$	2,5 + 19.922.944 + 1,25 = 19.922,947 ns

E/S per DMA:

N_{prog}	programar la transferència	40 instruccions
N_{final}	finalitzar la transferència	30 instruccions

t_{prog}	$N_{prog} \cdot t_{inst}$	$40 \cdot 0,125 = 5$ ns
t_{final}	$N_{final} \cdot t_{inst}$	$30 \cdot 0,125 = 3,75$ ns
t_{transf_dada}	$t_{cessió} + t_{mem} + t_{recup}$	0,5 + 1 + 0,5 = 2 ns
t_{transf_bloc}	$N_{dades} \cdot t_{transf\_dada}$	$2 \cdot 2^{20} \cdot 2 = 4.194.304$ ns = 4,19 ms
t_{cpu}	$t_{prog} + t_{transf\_bloc} + t_{final}$	5 + 4.194.304 + 3,75 = 4.194.312 ns

E/S per DMA en mode ràfega:

N_{prog}	programar la transferència	40 instruccions
N_{final}	finalitzar la transferència	30 instruccions
m_{buffer}	mida de la memòria interna en mode ràfega	32 bytes

t_{prog}	$N_{prog} \cdot t_{inst}$	$40 \cdot 0,125 = 5$ ns
t_{final}	$N_{final} \cdot t_{inst}$	$30 \cdot 0,125 = 3,75$ ns
$N_{ràfega}$	m_{buffer} / m_{dada}	32 / 4 = 8
$t_{transf\_ràfega}$	$t_{cessió} + (N_{ràfega} \cdot t_{mem}) + t_{recup}$	$0,5 + (8 \cdot 1) + 0,5 = 9$ ns
t_{transf_bloc}	$(N_{dades} / N_{ràfega}) \cdot t_{transf\_ràfega}$	$(2 \cdot 2^{20} / 8) \cdot 9 = 2.359.296$ ns = 2,35 ms
t_{cpu}	$t_{prog} + t_{transf\_bloc} + t_{final}$	5 + 2.359.296 + 3,75 = 2.359.304 ns

Comparem el temps d'ocupació del processador en cadascuna de les tècniques d'E/S.

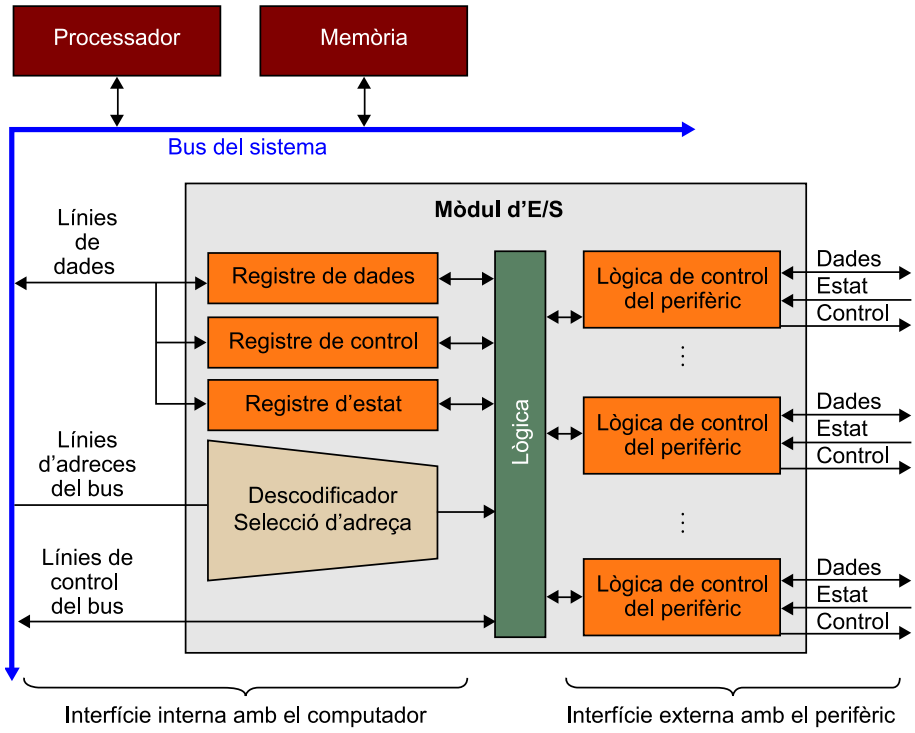
Perifèric:

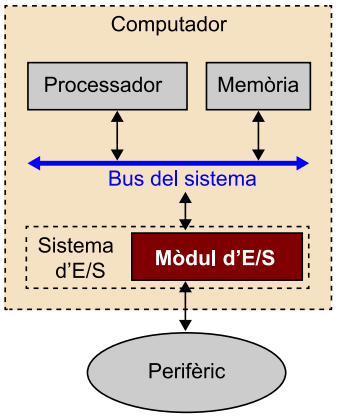
t_{bloc}	69,5 ms
$\%_{ocupació}$	$(t_{transf\_bloc} \cdot 100) / t_{bloc}$
t_{disp}	$t_{bloc} - t_{transf\_bloc}$

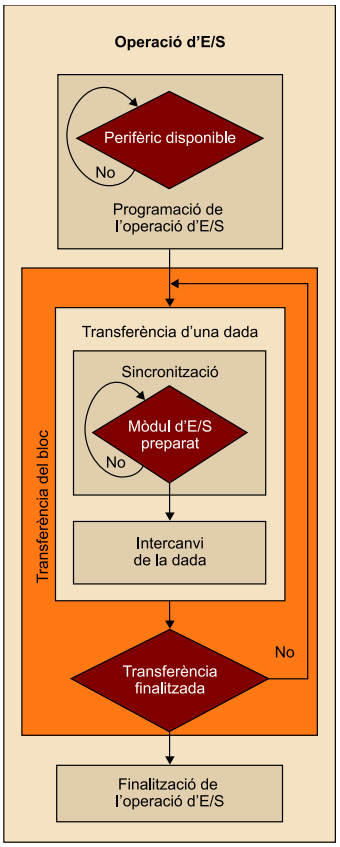
	E/S programada	E/S per interrupcions	E/S per DMA	E/S per DMA en mode ràfega
t_{transf_bloc}	69,5 ms	19,923 ms	4,194 ms	2,359 ms
$\%_{ocupació}$	100%	28,66%	6,03%	3,39%
t_{disp}	0 ms	49,577 ms	65,306 ms	67,141 ms

Resum

En aquest mòdul s'ha explicat en primer lloc els aspectes bàsics del sistema d'E/S d'un computador. L'estructura del sistema d'E/S formada pels perifèrics, els mòduls d'E/S i els sistemes d'interconnexió externs com a elements principals.







A continuació s'ha descrit les fases que componen una operació bàsica d'E/S.

1) Programació de l'operació d'E/S.
2) Transferència de dades.
3) Finalització de l'operació d'E/S.

Per a fer la transferència de dades s'ha descrit les principals tècniques d'E/S:

- E/S programada.
- E/S per interrupcions.
- E/S per DMA.

Per a cadascuna d'aquestes tècniques s'ha explicat el maquinari necessari, com funciona la transferència d'una dada individual i d'un bloc entre el computador i un perifèric, i també com s'ha de fer la gestió de la transferència quan tenim més d'un perifèric connectat al computador i s'ha de gestionar simultà-

niament múltiples transferències en què els problemes principals que cal resoldre són la identificació del perifèric amb el qual s'ha de fer la transferència i la gestió de prioritats.

Finalment s'ha fet una comparació de les diferents tècniques d'E/S per a analitzar les prestacions d'un sistema d'E/S i d'aquesta manera aprofundir una mica més en el funcionament d'aquestes tècniques.