

Scriere cu invalidare (1)

- un procesor modifică o dată
- modificarea se face în cache-ul propriu
 - toate celelalte cache-uri sunt notificate
- celelalte cache-uri
 - nu au o copie a datei modificate - nici o acțiune
 - au o copie a datei modificate - își invalidează linia corespunzătoare
 - valoarea corectă va fi preluată când va fi nevoie

Scriere cu invalidare (2)

x - variabilă partajată

Proc.	Acțiune	Reacție cache	Cache A	Cache B	Memorie
					9
A	i=x;	read miss	9		9
B	j=x;	read miss	9	9	9
A	x=5;	invalidation	5	inv.	9
B	k=x;	read miss	5	5	5

Scriere cu actualizare (1)

- un procesor modifică o dată
- modificarea se face în cache-ul propriu
 - toate celelalte cache-uri sunt notificate
 - se transmite noua valoare
- celelalte cache-uri
 - nu au o copie a datei modificate - nici o acțiune
 - au o copie a datei modificate - preiau noua valoare

Scriere cu actualizare (2)

x - variabilă partajată

Proc.	Acțiune	Reacție cache	Cache A	Cache B	Memorie
					9
A	i=x;	read miss	9		9
B	j=x;	read miss	9	9	9
A	x=5;	invalidation	5	5	5
B	k=x;	read hit	5	5	5

Invalidare vs. actualizare (1)

- mai multe scrieri succesive în aceeași locație
 - *write invalidation* - o singură invalidare (prima dată)
 - *write update* - câte o actualizare pentru fiecare scriere
 - mai avantajos - invalidare

Invalidare vs. actualizare (2)

- mai multe scrieri în aceeași linie de cache
 - modificarea unei locații necesită invalidarea/actualizarea întregii linii
 - *write invalidation* - o singură invalidare (prima dată)
 - *write update* - câte o actualizare pentru fiecare scriere
 - mai avantajos - invalidare

Invalidare vs. actualizare (3)

- "timpul de răspuns"
 - timpul între modificarea unei valori la un procesor și citirea noii valori la alt procesor
 - *write invalidation* - întâi invalidare, apoi citire (când este necesar)
 - *write update* - actualizare imediată
 - mai avantajos - actualizare

Invalidare vs. actualizare (4)

- ambele variante au avantaje și dezavantaje
- scrierea cu invalidare - ocupare (mult) mai redusă a memoriei și magistralelor
- scrierea cu actualizare - rata de succes a cache-urilor mai mare
- mai des folosită - invalidarea

III. Dispozitivele periferice

Dispozitiv periferic

- realizează o formă oarecare de comunicare între procesor și "lumea exterioară"
- pentru gestiunea comunicării cu procesorul, include un controller I/O
 - de obicei, acesta conține o serie de regiștri care rețin informații necesare pentru comunicare
 - date
 - informații de stare
 - comenzi (de la procesor)

Intrări-ieșiri (I/O)

- cum privește sistemul comunicarea
 - I/O proiectat pe memorie
 - citirile/scrierile sunt văzute ca și cum ar fi realizate în locații de memorie
 - adresele I/O - în spațiul de adrese de memorie
 - aceleași semnale de control ca pentru memorie
 - I/O izolate
 - adresele I/O - separate de adresele de memorie
 - semnale de control diferite de ale memoriei

Moduri de comunicare (1)

- intrări-ieșiri programate
 - *programmed I/O*
 - programul așteaptă într-o buclă până când perifericul inițiază un transfer
 - eficient dacă se știe dinainte momentul când perifericul va solicita comunicarea
 - consum inutil de timp de execuție al procesorului

Moduri de comunicare (2)

- acces direct la memorie
 - *Direct Memory Access* (DMA)
 - un controller specializat (DMA controller) efectuează transferurile
 - foarte rapid
 - preia controlul magistralelor și transferă datele direct între periferic și memorie
 - fără intervenția procesorului
 - util pentru transferul volumelor mari de date

Moduri de comunicare (3)

- intrări-ieșiri gestionate prin întreruperi
 - interrupt-driven I/O
 - când un periferic dorește să inițieze comunicarea, notifică procesorul
 - printr-o cerere de întrerupere
 - în restul timpului, procesorul poate îndeplini alte sarcini
 - metoda cea mai flexibilă

Magistrale (1)

- căi de comunicație a informației
- o magistrală leagă între ele mai mult de 2 componente printr-o cale unică
- descrierea unei magistrale
 - semnalele electrice folosite
 - reguli de comunicare - trebuie respectate de toate părțile implicate
 - mod de conectare

Magistrale (2)

Accesul la magistrală

- mai multe entități pot solicita simultan accesul
- este necesară o procedură de arbitraj
 - decide cine va primi accesul
 - celelalte trebuie să aștepte eliberarea magistralei

Arbitrarea magistralelor

Tipuri de arbitrare a magistralelor

- centralizat
 - decizia o ia un circuit specializat (arbitru)
- descentralizat
 - componentele se înțeleg între ele
 - pe baza regulilor care definesc funcționarea magistralei

Conectarea la magistrală

- probleme de natură electrică
- mai multe circuite conectate împreună
 - în intrare și ieșire
- nu se pot conecta mai multe ieșiri
 - nivelele diferite de tensiune ar distruge circuitele
- o soluție - multiplexarea
 - toate ieșirile sunt conectate la un multiplexor

Circuite *tri-state*

- ieșirea are 3 stări posibile
 - 0
 - 1
 - impedanță infinită (*High-Z*)
- primele două corespund valorilor obișnuite
- a treia implică decuplarea de pe magistrală
 - ca și cum ieșirea circuitului nici nu ar fi conectată la magistrală

Circuite *open-collector*

- în unele cazuri se numesc *open-drain*
 - în funcție de tehnologia utilizată
- este posibilă conectarea mai multor ieșiri simultan
- valoarea rezultată - funcția AND între ieșirile conectate

Magistrale - privire generală

Avantaje

- activitatea pe magistrală - ușor de controlat
- economic - structură relativ simplă

Dezavantaj

- performanțe mai scăzute
 - doar 2 componente pot comunica la un moment dat

IV. Sistemul de întreruperi

Ce este o întrerupere?

- procesorul poate suspenda execuția programului curent
- scop - tratarea unor situații neprevăzute
- după tratare se reia programul întrerupt
- scopul inițial - comunicarea cu perifericele
- procesorul nu "așteaptă" perifericele
 - acestea îl solicită când este necesar

Întreruperi hardware

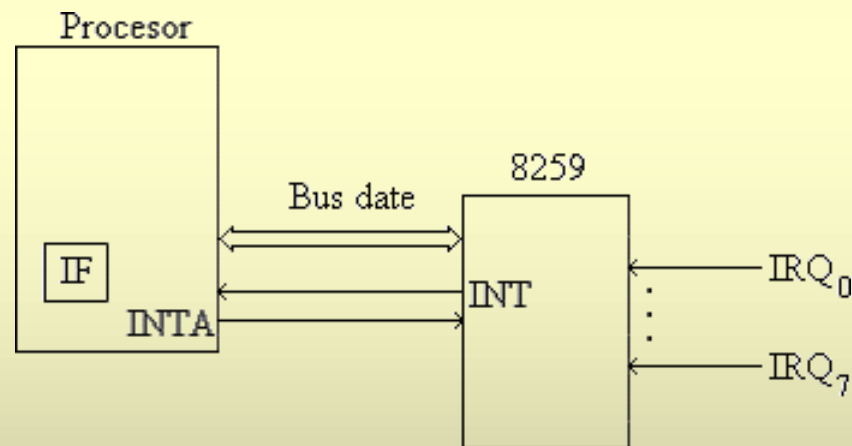
- dezactivabile (*maskable*)
 - procesorul poate refuza tratarea lor
 - depinde de valoarea bistabilului IF (*Interrupt Flag*): 1 - acceptare, 0 - refuz
 - IF poate fi modificat prin program
- nedezactivabile (*non-maskable*)
 - procesorul le tratează întotdeauna

Controllerul de întreruperi (1)

- circuit specializat
- preia cererile de întrerupere de la periferice
- le trimite procesorului
- rezolvă conflictele (mai multe cereri simultane) - arbitrare
 - fiecare periferic are o anumită prioritate

Controllerul de întreruperi (2)

- inițial - Intel 8259
 - se pot folosi mai multe (cascadare)



- astăzi - integrat în chipset

Tratarea întreruperilor - etape (1)

- dispozitivul periferic generează o cerere de întrerupere pe linia sa IRQ_i
- controllerul trimite un semnal pe linia INT
- procesorul verifică valoarea bistabilului IF
 - numai pentru întreruperi dezactivabile
 - dacă este 0 - refuză cererea; stop
 - dacă este 1 - răspunde cu un semnal INTA

Tratarea întreruperilor - etape (2)

- se întrerupe execuția programului curent
- se salvează în stivă regiștrii (inclusiv PC)
- se șterge bistabilul IF
 - se blochează execuția altei întreruperi în timpul execuției programului pentru întreruperea în curs
 - poate fi repus pe 1 prin program

Tratarea întreruperilor - etape (3)

- identificare periferic (sursa cererii)
 - controllerul depune pe magistrala de date un octet *type*
 - indică perifericul care a făcut cererea
 - maximum $2^8=256$ surse de întrerupere
 - fiecare sursă are rutina proprie de tratare
 - periferice diferite - tratări diferite

Tratarea întreruperilor - etape (4)

- determinarea adresei rutinei de tratare
 - adresa fizică 0 - tabelul vectorilor de întreruperi
 - conține adresele tuturor rutinelor de tratare
 - dimensiune: $256 \text{ adrese} \times 4 \text{ octeți} = 1 \text{ Ko}$
 - octet type = $n \rightarrow$ adresa rutinei de tratare: la adresa $n \times 4$

Tratarea întreruperilor - etape (5)

- salt la rutina de tratare a întreruperii
- execuția rutinei de tratare
- revenire în programul întrerupt
 - restabilire valoare bistabil IF
 - restabilire valoare regiștri (din stivă)
 - reluarea execuției programului de unde a fost întrerupt

Extindere

- sistem puternic și flexibil
- poate fi extins - utilizare mai largă
- programele trebuie întrerupte și în alte situații (nu doar pentru comunicarea cu perifericele)
- util mai ales pentru sistemul de operare

Tipuri de întreruperi

- *hardware* - generate de periferice
- *excepții (traps)* - generate de procesor
 - semnalează o situație anormală
 - exemplu: împărțire la 0
- *software* - generate de programe
 - folosite pentru a solicita anumite servicii sistemului de operare

V. Sistemul de operare

Rolul sistemului de operare (1)

- program cu rol de gestiune a sistemului de calcul
- face legătura între hardware și aplicații
- pune la dispoziția aplicațiilor diferite servicii
- supraveghează buna funcționare a aplicațiilor
 - poate interveni în cazurile de eroare

Rolul sistemului de operare (2)

- pentru a-și îndeplini sarcinile, are nevoie de suport hardware
 - cel mai important - sistemul de întreruperi
- componente principale
 - nucleu (*kernel*)
 - drivere

V.1. Nucleul sistemului de operare

Nucleul sistemului de operare

- în mare măsură independent de structura hardware pe care rulează
- "creierul" sistemului de operare
- gestiunea resurselor calculatorului - hardware și software
 - funcționare corectă
 - alocare echitabilă între aplicații

Moduri de lucru ale procesorului

- utilizator (*user mode*)
 - restricționat
 - accesul la memorie - numai anumite zone
 - accesul la periferice - interzis
- nucleu (*kernel mode*)
 - fără restricții

Modul de rulare al programelor

- sistemul de operare - în mod nucleu
 - poate efectua orice operație
- aplicațiile - în mod utilizator
 - nu pot realiza anumite acțiuni
 - apelează la nucleu

Trecerea între cele două moduri

- prin sistemul de întreruperi
- utilizator → nucleu
 - apel întrerupere software
 - generare excepție (eroare)
- nucleu → utilizator
 - revenire din rutina de tratare a întreruperii

Consecințe

- codul unei aplicații nu poate rula în modul nucleu
- avantaj: erorile unei aplicații nu afectează alte programe
 - aplicații
 - sistemul de operare
- dezavantaj: pierdere de performanță

Structura nucleului

- nu este un program unitar
- colecție de rutine care cooperează
- funcții principale
 - gestiunea proceselor
 - gestiunea memoriei
 - gestiunea sistemelor de fișiere
 - comunicarea între procese

V.2. Apeluri sistem

Apeluri sistem (*system calls*)

- cereri adresate nucleului de către aplicații
- acțiuni pe care aplicațiile nu le pot executa singure
 - numai în modul nucleu al procesorului
 - motiv - siguranța sistemului
- realizate prin întreruperi software
- similar apelurilor de funcții

Etapele unui apel sistem (1)

1. programul depune parametrii apelului sistem într-o anumită zonă de memorie
2. se generează o întrerupere software
 - procesorul trece în modul nucleu
3. se identifică serviciul cerut și se apelează rutina corespunzătoare

Etapele unui apel sistem (2)

4. rutina preia parametrii apelului și îi verifică
– dacă sunt erori - apelul eșuează
5. dacă nu sunt erori - realizează acțiunea cerută
6. terminarea rutinei - rezultatele obținute sunt depuse într-o zonă de memorie accesibilă aplicației care a făcut apelul

Etapele unui apel sistem (3)

7. procesorul revine în mod utilizator
8. se reia execuția programului din punctul în care a fost întrerupt
 - se utilizează informațiile memorate în acest scop la apariția întreruperii
9. programul poate prelua rezultatele apelului din zona în care au fost depuse

Apeluri sistem - concluzii

- comunicare între aplicație și nucleu
 - depunere parametri
 - preluare rezultate
- acțiunile critice sunt realizate într-un mod sigur
- mari consumatoare de timp
- apeluri cât mai rare - lucru cu buffere

Cum folosim bufferele

Exemplu - funcția *printf*

- formatează textul, apoi îl trimite spre ecran
- nu are acces direct la hardware
 - se folosește de un apel sistem
 - *write* (în Linux)
- de fapt, *printf* depune textul formatat într-o zonă proprie de memorie (buffer)
 - doar când bufferul e plin se face un apel sistem

V.3. Driveri

Ce sunt driverele?

- module de program care gestionează comunicarea cu perifericele
 - specializate - câte un driver pentru fiecare periferic
- parte a sistemului de operare
 - acces direct la hardware
 - se execută în modul nucleu al procesorului

Utilizare

- nu fac parte din nucleu
 - dar se află sub comanda nucleului
- folosite de rutinele de tratare ale întreruperilor hardware
- înlocuire periferic → înlocuire driver
 - organizare modulară
 - nu trebuie reinstalat tot sistemul de operare