Creșterea performanței CPU – tehnici:

* **Pipeline**:

Latența = nr cicluri de ceas necesar pt execuția unei instrucțiuni

Rata de execuție (throughput) = nr de instrucțiuni terminate într-un ciclu de ceas

Stagii = secvența pașilor prin care trece execuția unei instrucțiuni

* **Multiplicarea unităților de execuție** – mai multe ALU
* **Predicția salturilor** (statică/dinamică) – rezolvă dependențele de control
* **Execuția speculativă** - se execută toate variantele posibile înainte de a şti care este cea corectă
* **Predicația**:

Procesorul conţine regiştri de predicate (condiţii booleene), iar fiecare instrucţiune obişnuită are asociat un asemenea predicat. O instrucţiune produce efecte dacă şi numai dacă predicatul asociat este *true.*

* **Execuția out-of-order**:

Instrucţiunile nu se mai termină obligatoriu în ordinea în care şi-au început execuţia

* **Redenumirea regiștrilor** – rezolvă unele dependențe de date
* **Hyperthreading**:

Avem un singur procesor real, dar acesta apare ca două procesoare virtuale. Sunt duplicate componentele care reţin starea procesorului.

* **Arhitectura RISC** (*Reduced Instruction Set Computer) –* instrucțiuni mai simple și mai mulți regiștri

CISC (*Complex Instruction Set Computer*) – mai multe instrucțiuni și mai puțini regiștri

**Dependețe**:

* structurale: instrucţiuni aflate în stagii diferite au nevoie de aceeaşi componentă
* de date (RAW, WAR, WAW): o instrucţiune calculează un rezultat, alta îl foloseşte
* de control

Arhitecturi paralele de calcul: structuri pipeline, multiprocesor, distribuite

**Tipuri de sisteme multiprocesor**:

* cu memorie partajată centralizată:

Accesibilă tuturor procesoarelor, memorie comună, același timp de acces la memorie pentru orice procesor și orice locație.

* cu memorie partajată distribuită:

Spațiu de adrese unic, memorie distribuită fizic, acces neuniform la memorie.

* cu schimb de mesaje:

Fiecare procesor are propria memorie locală și nu e accesibilă celorlalte procesoare.

Actualizarea cache-urilor:

Un procesor modifică o dată și modificarea se face în cache-ul propriu, iar toate

celelalte cache-uri sunt notificate.

* **scriere cu invalidare**:

Dacă celelalte cache-uri au o copie a datei modificate, atunci îşi invalidează

linia corespunzătoare.

* **scriere cu actualizare**:

Dacă celelalte cache-uri au o copie a datei modificate, atunci preiau noua valoare.

Moduri de lucru ale procesorului:

* **user mode** (aplicațiile): restricționat, acces la memorie limitat, acces interzis la periferice
* **kernel mode** (sistemul de operare)

**Segmentarea memoriei**:

Segment = zonă continuă de memorie

Adresa unei locații = adresa de început a segmentului + deplasament

Descriptor = structură de date pt gestionarea unui segment

**Paginarea memoriei**:

Spațiul adreselor virtuale e împărțit în pagini.

Spațiul adreselor fizice e împărțit în cadre de pagină.

Catalogul de pagini conține adrese de tabele de pagini.

**Structura CPU**:

**ALU** (unitatea aritmetică și logică) – efectuează calculele propriu-zise

**Regiștri de uz general**

**Unitatea de control** – comandă celelalte componente și stabilește ordinea temporală

**Magistrala internă**

**PC** (contorul program) – reține adresa următoarei instrucțiuni de executat

**IR** (registrul de instrucțiuni) – reține codul ultimei instrucțiuni aduse din memorie

**Regiștri de interfață** (de adrese/date) – asigură comunicarea cu magistralele

**Regiștri temporari**

Rezolvare sesiune 2012:

3. Dintre tehnicile de creștere a performanței CPU, care rezolvă probleme ridicate de către dependențele de date?

R: Redenumirea regiștrilor.

4. Ce asemănări și deosebiri există între hyperthreading și superpipeline?

R: Hyperthreading exploatează mai bine structura pipeline. Când o instrucțiune a unui procesor virtual se blochează, celălalt procesor preia controlul. în pipeline, există probleme de dependență: o instrucțiune trebuie să aștepte terminarea alteia.

Asemănare: Ambele lucrează cu mai multe procese deodată.

5. Se consideră un sistem cu 2 procesoare, pe care se execută secvența de instrucțiuni de mai jos. Dacă variabilele x și y sunt partajate de procesele care rulează pe procesoare, indicați nr de ratări în cache-urile procesoarelor, în cazul în care se folosește scrierea cu invalidare. La începutul secvenței, variabilele partajate nu se găsesc în nici un cache.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 1 | 2 | 1 | 2 | 1 | 2 |
| a=x+3; | b=x/2+c; | y=3; | x=x-1; | m=x; | y=x; | n=y; | c=y; |

Procesor

Instrucțiune

R: 5 ratări

6. De ce trecerea între modurile nucleu și utilizator ale procesorului se poate face numai prin intermediul întreruperilor?

R: Pentru ca acțiunile critice să fie realizate într-un mod sigur, erorile unei aplicații să nu afecteze alte programe. Pentru că permisiunile pentru cele 2 moduri diferă.

7. Se consideră un sistem cu o memorie de 256 Mo, care utilizează segmentarea memoriei. Ca măsură a gradului de fragmentare a memoriei e folosită valoarea medie a zonelor libere. Se realizează următoarele operații de creare și eliberare de segmente:

|  |  |  |  |
| --- | --- | --- | --- |
| Nr crt | Segment | Operațiune | Dimensiune |
| 1 | S1 | alocare | 52 Mo |
| 2 | S2 | alocare | 45 Mo |
| 3 | S3 | alocare | 70 Mo |
| 4 | S2 | eliberare | - |

|  |  |  |  |
| --- | --- | --- | --- |
| Nr crt | Segment | Operațiune | Dimensiune |
| 5 | S4 | alocare | 40 Mo |
| 6 | S5 | alocare | 35 Mo |
| 7 | S3 | eliberare | - |
| 8 | S6 | alocare | 50 Mo |

Care e valoarea medie a zonelor libere, dacă se folosește algoritmul Best Fit?

R: 5+20+54 = 79/3

8. Orice fișier e memorat pe disc sub forma unor blocuri de dimesiune fixă. Deși e posibil ca blocurile care formează un fișier să fie răspândite aleator pe disc, gruparea acestora în aceeași zonă a discului (prin defragmentare) asigură un acces mai rapid la fișier. Ar fi avantajos ca, în mod similar, și paginile unui proces să fie grupate în memorie?

R: Tabele de pagini pe mai multe nivele. Catalog de pagini cu adrese de tabele de pagini. Nu e avantajos.

9. Se consideră un sistem pe 32 biți, care utilizează paginarea pe 2 nivele, cu dimensiunea paginilor de 4 Ko (1 Ko = 1024 octeți). Fiecare tabel de pagini are 16 elemente. Catalogul de pagini al unui proces (care conține adresele de început ale tabelelor de pagini) arată astfel:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 |
| 163840 | 221184 | 28672 | 131072 | 65536 |

indice

adresă început tabel de pagini

Dacă procesul încearcă să acceseze adresa virtuală 165000, care este adresa de început a tabelului de pagini care va fi consultat?

10. În sistemele cu memorie virtuală, tabelele de paginare conțin pentru fiecare pagină și un bit special (dirty bit); acesta are valoarea 0 atunci când pagina din memorie și copia sa din fișierul de paginare sunt identice, respectiv 1 când există diferențe, ca urmare a operațiilor de scriere. Ce valoare trebuie să primească dirty bit la crearea paginii (atunci când este lansat procesul în execuție)?

R: 0, pentru că se resetează la aducerea paginii în memorie

Rezolvare sesiune 2008 – Nr 1:

1. Pe care dintre magistralele unui sistem de calcul sunt transmise codurile instrucțiunilor pe care le execută procesorul?

R: Magistrala de control/ magistrala internă.

2. Se consideră un sistem cu 32 Mo de memorie, cu dimensiunea paginii de 4 Ko (1 Ko = 1024 octeți). Două procese aflate simultan în memorie necesită 10620 octeți, respectiv 8230 octeți. Dintre acestea, 1000 octeți reprezintă o zonă de memorie partajată de cele două procese (se regăsește în tabelele de paginare ale ambelor procese). Câtă memorie se pierde prin fragmentare internă?

3. Dacă un procesor are un pipeline de 22 stagii, iar verificarea condiției de salt se face în stagiul 16, câte cicluri de ceas se pierd în cazul unei predicții greșite?

4. Prin ce diferă **arhitectura RISC** de cea CISC? Ce avantaje prezintă prima față de cea de-a doua?

R: **Arhitectura CISC** are un nr mare de instrucțiuni cu complexitate mai mare. Deci timpul de execuție e mare. Există un număr mic de regiștri, deci se accesează intensiv memoria. Multe instrucțiuni sunt rar folosite. Arhitectura RISC are un set de instrucțiuni simplificat și un nr mai mare de regiștri. Deci se reduce nr de accese la memorie, care sunt mai simple și cu mai puține blocaje.

8. Procesoarele RISC au un număr mare de registri de uz general, pentru a reduce numărul de accese la memorie. În condițiile sporului de performanță adus de utilizarea cache-ului, mai este justificată această abordare?

R: Da, regiștrii sunt mai rapizi decât cache-ul.

9. Descrieți părțile componente ale unui **fișier obiect**.

R: **Antetul**, cu informații de identificare și informații despre celelalte părți ale fișierului.

**Tabela punctelor de intrare**, cu numele simbolurilor (variabile și funcții) din modulul curent care pot fi apelate din alte module.

**Tabela referințelor externe**, cu numele simbolurilor din alte module, dar utilizate în modulul curent.

**Codul** propriu-zis, rezultat din compilare și care va apărea în fișierul executabil.

**Dicționarul de relocare**, cu informații despre localizarea instrucțiunilor din partea de cod care necesită modificarea adreselor cu care lucrează.

10. Un proces aflat în execuție este scos din această stare la terminarea perioadei de timp alocate (chiar dacă nu a generat nici o eroare si nu a solicitat efectuarea nici unui apel sistem), pentru a ceda locul altui proces. În general, toate procesele au alocate perioade de execuție egale. Ar putea fi justificată ideea ca unele procese să aibă alocate perioade de execuție mai lungi decât altele?

Rezolvare sesiune 2008 – Nr 2:

5. De ce registrul PC (contorul program) nu poate fi modificat direct prin program?

Rezolvare sesiune 2008 – Nr 3:

1. Se consideră un sistem cu o memorie de 175000 octeți, care foloseste segmentarea memoriei. La un moment dat, tabelul descriptorilor de segment arată astfel (s-au reprezentat doar segmentele alocate):

Indice 1 2 5 7 8

Adresă start 25000 1000 150000 111000 83000

Dimensiune 24000 14000 15000 23000 25000

Mai trebuie alocate două segmente, de 9000 octeți, respectiv 30000 octeți (în această ordine). Dintre

algoritmii de plasare a segmentelor în memorie, care reușesc plasarea ambelor segmente și care eșuează? Nu se face compactarea memoriei.

R: First Fit & Best Fit – reușesc, Worst Fit – eșuează

3. Ce limitări prezintă modul utilizator al procesorului și de ce au fost introduse aceste limitări?

R: Accesul la memorie se poate face numai pe anumite zone, iar accesul la periferice e interzis. Pentru siguranța sistemului.

4. În ce situații este de preferat **legarea dinamică** a unei funcții în locul legării statice?

R: Pentru proceduri care tratează situații excepționale și proceduri folosite de mai multe programe.

5. Ce acțiuni realizează **editorul de legături** cu ajutorul informațiilor din fișierele obiect?

R: Editorul de legături construiește o tabelă cu toate modulele obiect și dimensiunile acestora. Atribuie adrese de start modulelor obiect. Determină instrucțiunile care realizează accese la memorie și adună la fiecare adresă o constantă de relocare. Determină instrucțunile care apelează funcții sau date din alte module și inserează adresele corespunzătoare.

6. Ce este o **dependență structurală** într-un sistem cu pipeline? Dați un exemplu.

R: O dependență structurală este atunci când instrucțiuni aflate în stagii diferite au nevoie de aceeași componentă. De exemplu, accesele la memorie (citire cod instrucțiune, citire operand, scriere rezultat) sau ALU (instrucțiuni aritmetice, calcului adreselor operanzilor, actualizarea valorii PC).

10. Dacă primul Ko de memorie ar fi ocupat de memorie ROM, adresele rutinelor de tratare ale

întreruperilor nu ar putea fi modificate. Din punct de vedere al sistemului de operare, această abordare ar fi avantajoasă sau dezavantajoasă?

9. Dacă ar exista un singur tabel de paginare global, în loc de a avea câte un tabel pentru fiecare proces, s-ar ocupa mai putin spațiu cu aceste structuri. Ce dezavantaj ar avea o asemenea soluție?

R: Timp.