Ciclul "Fetch-Execute"

Funcționarea calculatorului implică în fiecare moment anumite activități prin care acesta execută diverse comenzi primite de la sistemul de operare sau de la programele utilitare, comenzi ce implică de fiecare dată execuția unor instrucțiuni. Operația de bază a unui procesor este reprezentată de așa-numitul ciclu "Fetch-Decode-Execute" sau, pe scurt, "Fetch-Execute". Pe scurt, această operație implică faptul că pentru a executa o instrucțiune, de fiecare dată, procesorul citește o instrucțiune de program din memorie, o decodifică și apoi o execută. Acest lucru poate părea anevoios dar viteza fantastică la care acționează microprocesorul face ca procedeul de aducere din memorie, decodificare și apoi execuție să se desfășoare extrem de rapid.

Cei trei „actori" implicați în acest proces, microprocesorul, memoria și magistrala de memorie acționează la diferite viteze, de aceea, cu cât una dintre componente are o viteză de acționare mai mare, cu atât va contribui mai mult la scurtarea timpului de execuție a unui program. Există aplicații ce solicită intens lucrul cu memoria – în acest caz memoriile ieftine și rapide prezintă un avantaj, în timp ce alte aplicații pot solicita intens activitatea procesorului – desigur că în acest caz un procesor mai rapid va crește performanțele de calcul.

Din punct de vedere al tipului de memorie principală folosit de către calculatoare, există în principiu două tipuri de bază: cipuri DRAM (Dynamic Random Access Memory), care nu sunt atât de rapide precum procesorul. Un alt tip de memorie este memoria SRAM (Static Random Access Memory), care funcționează la viteze mai mari dar are și costuri de producție mult mai ridicate; astfel de cipuri de memorii sunt utilizate la memoriile *cache*. Memoriile cache sunt utilizate pentru a reduce decalajul dintre viteza microprocesorului și aceea a memoriei principale prin copierea și stocarea instrucțiunilor și datelor ce vor fi folosite imediat de către procesor.

Procesoarele RISC și chiar procesoarele CISC (Intel) au introdus o serie de tehnici noi pentru a mări eficiența ciclului „Fetch-Execute" prin procesarea simultană a mai multor instrucțiuni; cea mai cunoscută astfel de tehnică fiind tehnica prelucrării în conductă *(pipeline)* a instrucțiunilor.

Ciclul fetch-execute

Computerul trebuie să citească și să se supună fiecărui program (inclusiv sistemului de operare), instrucțiune cu instrucțiune. La prima vedere este un handicap din punct de vedere al performanțelor dacă este comparat cu organismele vii capabile să desfășoare mii de activități simultan.

Operația de bază pe care o îndeplinește procesorul este operația fetch-execute, secvență în care fiecare instrucțiune din cadrul unui program este citită din memorie asociată programului în CPU, este decodificată și apoi executată. Doar viteza extraordinară a echipamentelor electronice face ca acest ciclu obositor care se repetă încontinuu să fie de reală valoare.

În orice fază a dezvoltării computerelor, una dintre cele trei unități implicate în acest ciclu – memoria, magistrala şi CPU a fost factorul limitativ. Acest lucru poate afecta atât parametrii cu care trebuie să lucreze inginerii şi, de asemenea, selectarea algoritmilor pentru rezolvarea problemelor. De exemplu, există uneori disponibile metode "memory-intensive" sau alteori metode "compute-intensive".

Dacă memoria este rapidă şi ieftină, atunci primele metode sunt de preferat; în celălalt caz este preferată a doua soluție. În prezent, cipurile DRAM de memorie nu sunt la fel de rapide precum CPU. Există disponibile şi cipuri mai rapide SRAM, dar care sunt mult mai scumpe şi de aceea sunt utilizate în mici *buffere* foarte rapide denumite şi memorii *cache*. Acest tip de memorii poate ajuta într-o oarecare măsură la reducerea diferenței dintre întârzierile cauzate de accesul la memorie prin stocarea unor cópii ale instrucțiunilor şi datelor curente.

Pentru a reduce şi mai mult gâtuirea obținută datorată efectului negativ a structurii secvențiale von Neumann, noile procesoare RISC măresc viteza de execuție a ciclului fetch-execute prin execuția simultană a (~5) instrucțiuni prin intermediul unei metode pipeline.

Oricât de familiari am fi cu modalitatea de lucru a computerelor, este practic imposibil să înțelegem viteza incredibilă la care acestea operează. Tabelul următor ne prezintă diferențele între viteze ale operațiunilor uzuale din lumea reală.

ns= 1 / 1000 000 000 s	**μs = 1 / 1 000 000 s**	**ms = 1 / 1 000 s**
Ciclul fetch/execute – 10 ns	Viteza luminii – 300 m/μs	Viteza de reacție umană – 300 ms
Funcționarea unei porți logice – 5 ns	Linie de scan TV – 60 μs	Cadru TV – 20 ms
Acces la memoria SRAM – 15 ns	Întrerupere – 2-10 μs	Acces la hard-disk – 10 ms
	Scânteie la motor – 10 μs	Rotație completă a motorului la maşină (la 3000 rpm) – 20 ms

Tabelul 1. Exemple de activități şi viteza asociată acestora

Din tabel ne putem da seama de diferența existentă între viteza de operare a calculatorului şi diferite viteze din lumea înconjurătoare. Lucrurile ce par extrem de rapide, cum ar fi liniile de scan TV, sunt de sute de ori mai încete decât un ciclu fetch-execute realizat de către microprocesor. După cum ştim, la cel mai de jos nivel programele în calculator sunt formate din şiruri de biți ce reprezintă codificarea binară a unor instrucțiuni, precum:

1000 1011 1110 1100

Acest şir de biți este echivalent cu următoarea instrucțiune a procesorului Pentium:

MOV BP , SP

Rezultatul instrucțiunii anterioare este acela de a copia conținutul registrului SP în registrul BP.

De asemenea, şirul de biți:

1000 1011 0011 0100 0001 0010

este echivalent cu instrucţiunea următoare în limbaj de asamblare (pentru Pentium):

MOV AX, 1234H

Sau, în limbajul C:

x = 4660;

Se observă astfel avantajul unui limbaj de programare de nivel înalt (în cazul nostru limbajul C) față de exprimarea în limbaj cod-maşină (şiruri de biți) sau în limbaj de asamblare. Instrucțiunea de mai sus va copia valoarea zecimală 4660 (*1234* în hexazecimal) în registrul AX (denumit şi registrul *acumulator*).

Ciclul fetch-execute reprezintă procesul prin care microprocesorul preia din memoria în care este stocată programul următoarea instrucțiune ce va fi executată, o decodifică şi execută operația pe care această instrucțiune o reprezintă.

În continuare prezentăm acest funcționarea generală a acestui proces. În figura 2 putem vedea microprocesorul cu registrul pointer de instrucţiune - IP *(Instruction Pointer)* şi registrul acumulator AX *(Accumulator Register).* Memoria principală stochează programul aflat în execuție, în care toate instrucțiunile de genul: **MOV BP,SP** sau **MOV AX, 1234H** se află reprezentate sub formă binară. Un registru de acces la memorie este utilizat pentru a putea accesa porțiunea de memorie de unde sunt preluate datele. Registrul pointer de instrucțiune indică tot timpul adresa din memorie a următoarei instrucțiuni ce va fi executată.

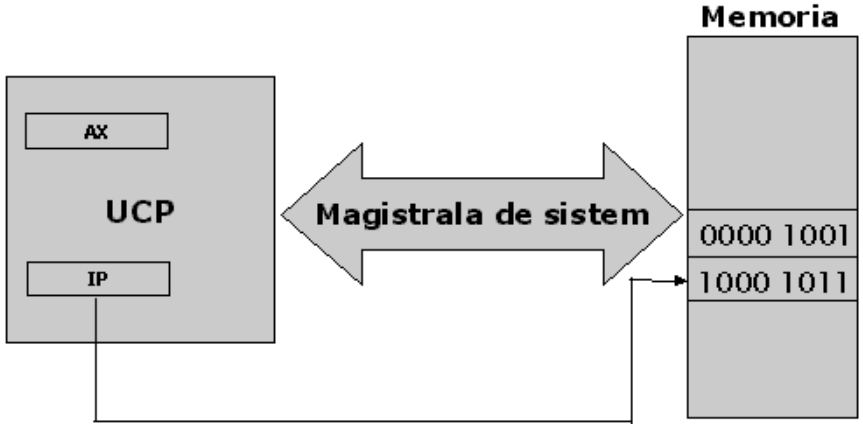

Figura 2. IP (Pointerul de instrucţiune) indică întotdeauna adresa următoarei instrucțiuni ce va fi executată

Figurile 3 şi 4 ne înfățișează procesul de preluare a datelor din memorie (etapa *fetch*). Această etapă este aproximativ identică pentru toate tipurile de instrucțiuni. Paşii urmați de microprocesor în această etapă sunt:

a) Adresa din registrul pointer de instrucțiune (IP) este copiată pe magistrala de adrese de memorie de unde este transmisă în registrul RAM (Registrul de Acces la Memorie);

b) Pointerul de instrucțiune este incrementat (IP++), indicând adresa de memorie a următoarei instrucțiuni ce va fi executată;

c) Se selectează locația de memorie şi se copiază conținutul acesteia în magistrala de date;

d) Procesorul copiază codul instrucțiunii din magistrala de date în registrul de instrucțiune;

e) Începe procesul de decodificare a instrucțiunii.

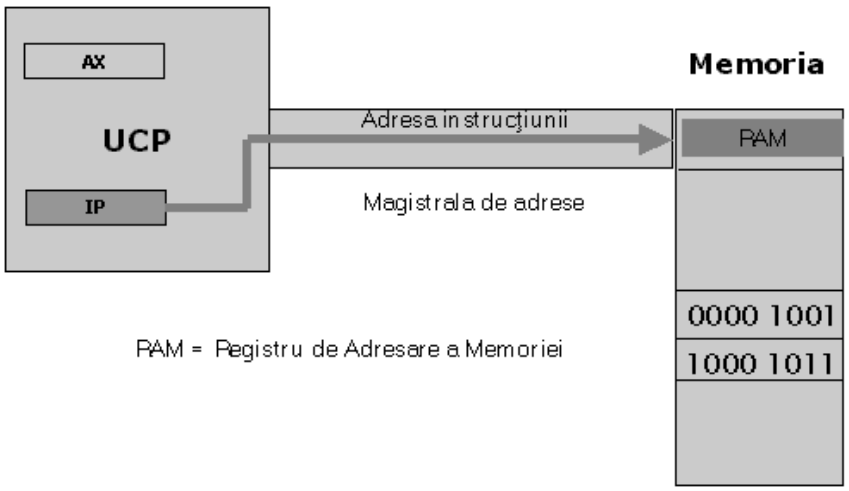

Figura 3. Ilustrarea pasului a) din ciclul *fetch*

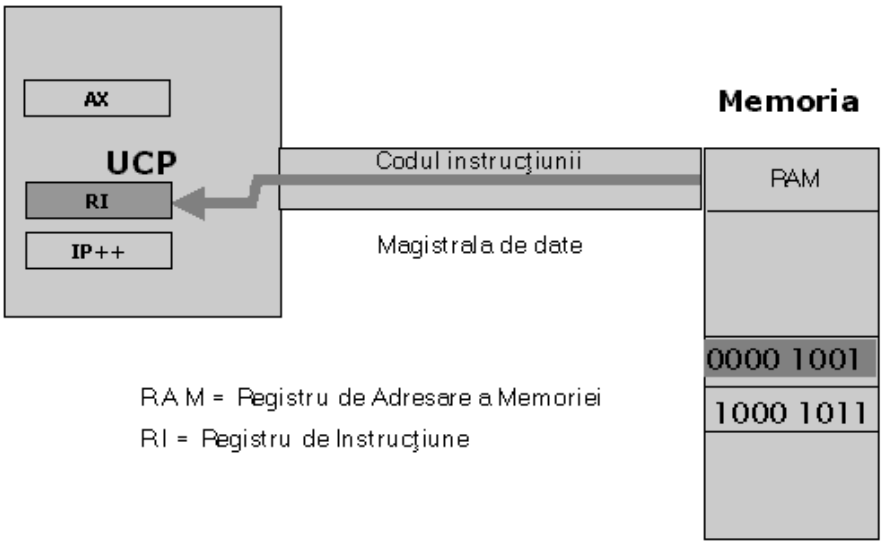

Figura 4. Ilustrarea paşilor b), c) şi d) din ciclul f*etch*

Figurile 5 şi 6 ne înfăţişează procesul de execuţie a instrucţiunii (etapa *execute*) în cazul instrucţiunii Intel **MOV AX, 1234H** (etapa *execute* diferă de la instrucţiune la instrucţiune). Paşii urmaţi de microprocesor în această etapă sunt:

a) Conţinutul registrului pointer de instrucţiune (IP) este copiat pe magistrala de adrese de memorie de unde este transmis în registrul RAM (Registrul de Acces la Memorie);

b) Pointerul de instrucţiune este incrementat (IP++);

c) Valoarea selectată din memorie (1234H) este copiată pe magistrala de date;

d) Procesorul copiază valoarea de pe magistrala de date în registrul AX.

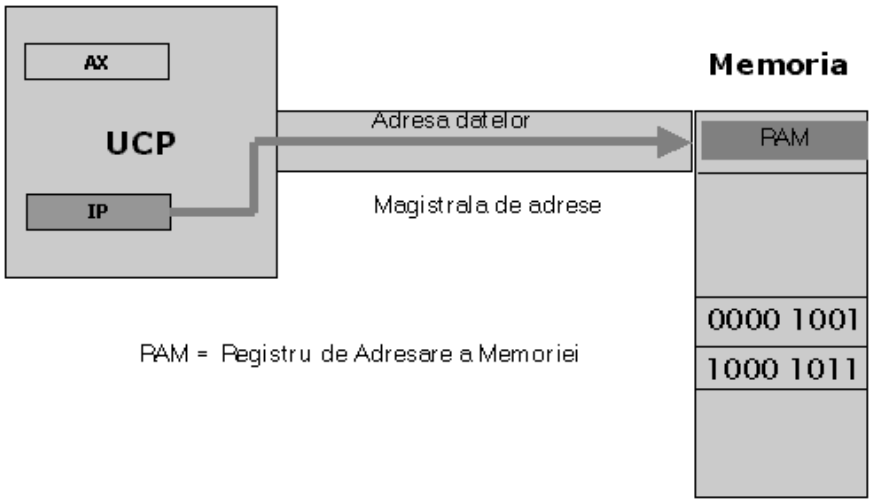

Figura 5. Ilustrarea pasului a) din ciclul *execute*

Acest proces reprezintă, de fapt, o simplificare a procesului *fetch-execute* ce se desfăşoară în cadrul unui microprocesor modern din zilele noastre. Unele instrucţiuni au nevoie de un ciclu de execuţie în plus pentru a citi valoarea unei

adrese din memorie, care este mai apoi folosită pentru a accesa valoarea variabilei respective, valoare stocată tot în memorie. Ca o concluzie, ciclul fetch-execute reprezintă secvența prin care fiecare instrucțiune a unui program este citită din memorie, decodificată şi apoi executată. Acest proces poate presupune ulterior mai multe sub-procese, cum ar fi citirea de date suplimentare din memorie şi stocarea rezultatelor operației/operațiilor înapoi în memorie.

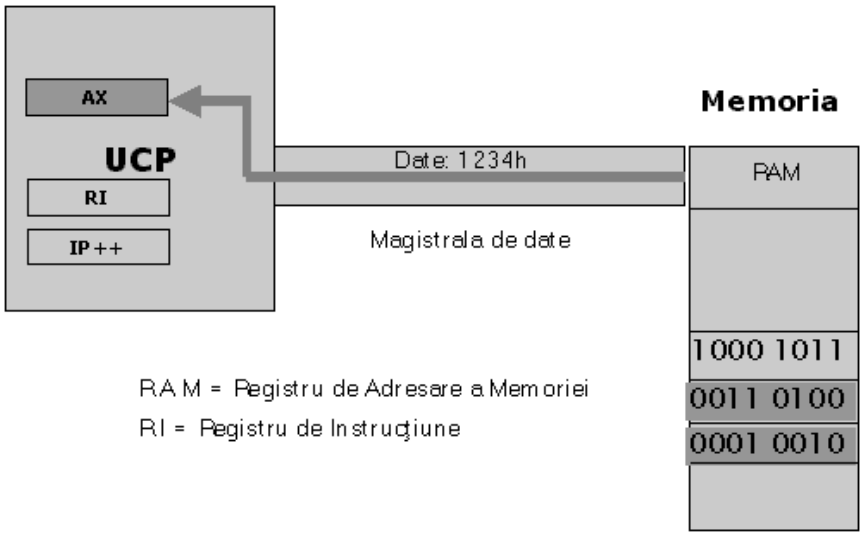

Figura 6. Ilustrarea paşilor b), c) şi d) din ciclul *execute*

Atât sistemul de operare Windows NT/2000 cât şi sistemul de operare UNIX oferă instrumente prin care pot fi vizualizate activitățile ce se află în lucru. Pe stațiile Sun cu sistemul de operare UNIX (varianta Solaris) există utilitarul denumit *perfmeter*, pentru Linux există *xsysinfo şi gsysinfo* iar pentru Windows NT există *Performance Monitor,* care a fost înlocuit în Windows 2000 de *System Monitor*. Prezentăm în continuare caracteristicile acestor aplicații utilizate pentru monitorizarea utilizării resurselor calculatorului.

1. UNIX – *perfmeter*

Pentru o stație Sun, comanda care ne arată detaliat activitatea microprocesorului este:

$ perfmeter –t cpu &

2. Linux

xsysinfo

Conform paginii de manual, utilitarul *xsysinfo* afişează parametrii kernelului sub formă grafică. Sintaxa completă este următoarea:

xsysinfo [-help] [-update *n*] [-[no]title] [-[no]labels] [-[no]loadavg] [-[no]load] [-[no]mem] [-[no]swap]

Pe scurt, *xsysinfo* este o aplicaţie XWindow folosită pentru afişarea unor parametri ai kernelului Linux în format grafic, o combinaţie a comenzilor *top*, *free* şi *xload*, cu diferenţa că valorile afişate (media gradului de utilizare a procesorului, gradul de utilizare a procesorului, dimensiunea de *swap*) sunt prezentate într-o fereastră orizontală.

Utilitarul *xsysinfo* afişează următoarele valori:

- *gradul mediu de utilizare al UCP* – valoarea afişată este între 0.000 şi 8.000. Bara orizontală a afişajului este împărţită în segmente, unde fiecare segment reprezintă o valoare de 1.

- *gradul de utilizare al UCP* – se afişează gradul de utilizare şi timpii de neutilizare pe trei segmente: utilizator, sistem şi nice. În cazul unui sistem multi-procesor opţiunea –*smp* afişează câte o bară orizontală pentru fiecare procesor în parte.

- *memoria* – bara grafică ce corespunde memoriei este împărţită în două segmente ce reprezintă dimensiunea fizică a memoriei calculatorului ce este utilizată de procese (în partea stângă) şi memoria utilizată pentru paginare şi memoria cache buffer în partea dreaptă. Întreaga lungime a segmentului orizontal ne arată memoria fizică utilizată de către sistem la momentul respectiv.

- *porţiunea de swap* – indică dimensiunea de spaţiu swap utilizat de către sistem din totalul spaţiului swap alocat.

gsysinfo

Gsysinfo este un utilitar conceput pentru interfaţa grafică Gnome şi urmăreşte utilitatea programului xsysinfo. Gsysinfo este conceput sub licenţă GNU – GPL (General Public Licence).

Referitor la numele de GNU, acesta provine de la sintagma „GNU Not UNIX" şi s-a dorit a fi un sistem de operare precum UNIX ce este distribuit cu codul sursă şi poate fi copiat, modificat şi redistribuit. Proiectul GNU a fost iniţiat în 1983 de Richard Stallman şi alţii ce au pus bazele Fundaţiei pentru Software Liber *(FSF – Free Software Foundation)*. Concepţia lui Stallman este aceea că utilizatorii pot face ce doresc cu software-ul achiziţionat, putând face cópii ale acestuia pentru prieteni şi modifica codul sursă redistribuind-ul la un anumit cost. FSF stipulează termenul *copyleft* care înseamnă că oricine redistribuie software *free* trebuie să lase în continuare libertatea de copiere şi redistribuţie a programului, asigurându-se în acest fel că nimeni nu va reclama drepturi de proprietate asupra unor versiuni viitoare şi nu va impune restricţii la utilizarea acestuia.

În acest context, termenul *free* înseamnă *libertate* şi nu neapărat *gratis*. Fundaţia FSF percepe nişte costuri iniţiale la distribuţia GNU. Redistribuitorii pot, de asemenea, să perceapă taxe pentru copiile programelor în scopul profitului sau pentru acoperirea costurilor. Ideea de bază a *software-ului liber* (*free software*) este aceea

că se lasă libertatea utilizatorilor să modifice şi să reasambleze produsul fără nici o restricţie în afară de aceea că nici ei, la rândul lor, nu pot impune restricţii mai departe.

Stallman crede că unul dintre rezultatele filozofiei *free software* este acela că mai multe programe *free* vor coexista împreună provenind din alte programe *free*. GNU[•] este un exemplu în acest sens; acesta a devenit un sistem de operare când în august 1996 i-a fost adăugat un kernel (GNU Hurd şi Mach). Fundaţia FSF continuă să dezvolte software *free* sub formă de programe de aplicaţii; un program de tip *spreadsheet* este acum disponibil. Sistemul de operare Linux este conceput cu componente GNU iar kernelul este dezvoltat de Linus Torvalds.

Capturi de ecran *gsysinfo*

Prezentăm în continuare câteva capturi de ecran gsysinfo.

- putem vedea în imaginea de mai jos utilitarul gsysinfo situat între bara de volum şi ceas (încărcarea sistemului este de aproximativ 1.8):

- în continuare apare indicatorul sysinfo pentru activitatea în reţea. Traficul de date transmise de la calculator spre reţea apare în culoarea verde iar traficul de intrare este ilustrat în roşu.

- în figurile 7, 8 şi 9 apar ferestrele legate de setarea caracteristicilor programului *gsysinfo*.

[•] GNU s-a vrut iniţial să fie o alternativă la versiunile comerciale de UNIX. Acest lucru nu s-a întâmplat încă, dar Richard Stallman şi alţi programatori muncesc în continuare pentru acest ideal. Paradoxal este că primele succese înregistrate de GNU au fost aplicaţii adiţionale sistemelor proprietare UNIX. Componente GNU precum GNU Emacs, GCC (GNU C Compiler) şi bash (un înlocuitor *free* pentru Bourne Shell) sunt instalate astăzi implicit pe majoritatea variantelor de UNIX existente.

Figura 7. Fereastra de setări pentru *gsysinfo* – caracteristici generale

Figura 8. Fereastra de setări pentru *gsysinfo* – indicatori

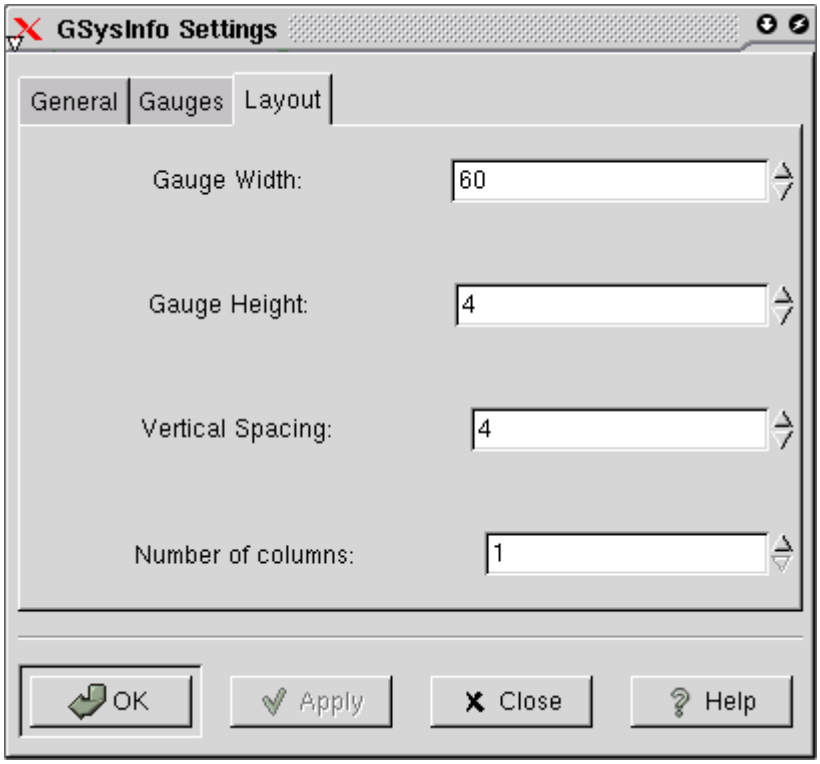

Figura 9. Fereastra de setări pentru *gsysinfo* – Layout

Windows 2000 - System Monitor

În Windows 2000, cu ajutorul lui System Monitor se pot măsura performanţele calculatorului local sau ale altor calculatoare din reţea. Utilitarul System Monitor asigură următoarele funcţionalităţi:

- Colecţionează şi vizualizează în timp real date legate de performanţa calculatorului local sau pentru alte calculatoare de la distanţă;
- Vizualizează datele colectate în timp real sau stocate anterior;
- Reprezintă datele sub formă de: grafic, histogramă sau raport de vizualizare;
- Încorporează funcţionalităţi ale aplicaţiei System Monitor în Microsoft Word sau alte aplicaţii ale suitei Microsoft Office cu ajutorul caracteristicei denumite *Automation*;
- Creează pagini HTML pentru vizualizarea performanţelor;
- Creează configuraţii de monitorizare reutilizabile ce pot fi instalate pe alte calculatoare ce folosesc MMC (Microsoft Management Console).

Cu ajutorul lui System Monitor se pot colecta şi vizualiza date legate de gradul de utilizare a componentelor hardware precum şi date legate de activităţile serviciilor

de sistem existente pe calculatoarele administrate. În cadrul aplicației se poate stabili modalitatea de prezentare a datelor în următoarele moduri:

- *Tipul de date* – pentru a selecta datele ce vor fi colectate, se pot specifica unul sau mai multe instanțe de contorizare pentru obiecte ale monitorului de performanțe. Unele obiecte (memoria, spre exemplu) oferă contorizare la nivelul resurselor sistemului; altele oferă posibilitate de contorizare la nivelul execuției aplicațiilor.
- *Sursa de date* – Aplicația System Monitor poate strânge date de pe calculatorul local sau de pe alte calculatoare din rețea unde există această permisiune (implicit este nevoie de drepturi de administrator). În plus, se pot include atât date culese în timp real cât și date stocate anterior în fișiere speciale de tip *log*.
- *Parametri de test* – se oferă posibilitatea de stabilire manuală, la cerere sau automată într-un interval specificat a unor teste de date. Atunci când se vizualizează aceste date se poate alege momentul de început sau de sfârșit astfel încât datele pot fi vizualizate între intervale specificate de timp.

Magistrala de sistem

Pentru a coordona și controla întrega activitate a calculatorului, microprocesorul trimite niște mesaje, denumite *semnale,* către componentele acestuia. Din punct de vedere al tipului acestor semnale, ele se pot clasifica în semnale pentru magistrala de date (de regulă cu o dimensiune de 32 sau 64 de biți), magistrala de adrese (de asemenea, pe 32 biți sau mai mult) și magistrala de control (formată din aproximativ 15 „linii" de control ce au rolul de a iniția sau stopa diverse activități din interiorul computerului). Una dintre liniile de control este reprezentată de ceasul de sistem, care este un oscilator de cristal de înaltă frecvență (pe placa de bază îl identificăm ca fiind un mic cilindru argintiu, situat în apropierea microprocesorului).

După cum am mai arătat, prin intermediul liniilor de magistrală (pentru a controla diversele activități ale calculatorului), microprocesorul trimite semnale către componentele calculatorului, componentele trimițând, la rândul lor, un răspuns către microprocesor. În unele cazuri o astfel de acțiune este controlată de un alt dispozitiv decât microprocesorul, care poate lua controlul asupra liniilor de magistrală (în acest mod se „eliberează" și procesorul de sarcina respectivă). Secvența semnalelor trimise prin intermediul magistralei trebuie să fie coordonate extrem de precis în timp printr-o acțiune de *sincronizare*. Dacă această operație de sincronizare este asigurată în întregime de către ceasul de sistem, atunci magistrala se numește sincronă.