ARHITECTURA CALCULATOARELOR 2003/2004
CURSUL 7

2.5. Instrucţiuni aritmetice si logice

Aşa cum am arătat mai devreme, grupul operaţilor aritmetice si logice conţine instrucţiuni care solicită operaţii aritmetice, logice sau de deplasare. În acest subcapitol vom studia mai în amânunt aceste operaţii.

2.5.1 Operaţii logice

În subcapitolul 1 am prezentat operaţiile logice AND (ŞI), OR (SAU) si XOR (SAU EXCLUSIV) ca pe nişte operaţii care combina 2 intrări de un bit pentru a produce o ieşire pe un bit. Aceste operaţii pot fi extinse la operaţii care combina 2 şiruri de biţi pentru a produce o ieşire de forma unui şir de biţi, aplicând operaţia elementara bit cu bit. De exemplu efectuarea unui AND intre cuvintele binare 10011010 si 11001001 produce ca rezultat :

```
    10011010
AND 11001001
    10001000
```

unde rezultatul s-a obţinut scriind la baza fiecărei coloane rezultatul operaţiei AND între cei doi biţi din coloană. Similar, operaţiile OR şi XOR între aceleaşi şiruri de biţi vor produce rezultatele:

```
    10011010          10011010
OR  11001001      XOR 11001001
    11011011          01010011
```

Operaţia AND este utilizată în principal pentru plasarea de biţi 0 într-o zonă a unui cuvânt binar fără a modifica restul. Să vedem, de exemplu, ce se întâmplă dacă octetul 00001111 este primul operand al operaţiei AND. Fără să cunoaştem conţinutul celui de-al doilea operand, vom putea trage concluzia că cei mai semnificativi 4 biţi ai rezultatului vor avea valoarea 0. În plus, cei mai puţin semnificativi 4 biţi ai rezultatului sunt o copie a aceleiaşi zone din cel de-al doilea operand, după cum observaţi şi din exemplul următor :

```
    00001111
AND 10101010
    00001010
```

Aceasta utilizare a operaţiei AND este un exemplu al procesului denumit mascare (masking). În acest caz un operand, denumit mască (mask), determină care parte a celuilalt operand va afecta rezultatul. Pentru operaţia AND, mascarea produce un rezultat care este o replica parţiala a unuia dintre operanzi, poziţiile neduplicate având valoarea 0.

O asemenea operație este utilă atunci când este manipulată o harta de biți (bit map), un şir în care fiecare bit reprezintă prezența sau absența unui obiect. De exemplu, un şir de 52 de biți, în care fiecare bit este asociat unei cărți de joc, poate fi utilizat pentru descrierea unei mâini de poker atribuind valoarea 1 celor 5 biți corespunzători cărților de joc primite si 0 celorlalți biți. Similar, un şir de 52 de biți, in care 13 biți sunt 1 poate fi utilizat pentru a reprezenta o mână la jocul de bridge, iar un şir de 32 de biți poate fi utilizat pentru a preciza care dintre cele 32 de arome de înghețată sunt disponibile.

Să presupunem că o celulă de memorie de 8 biți este utilizată ca bitmap şi că vrem să vedem dacă este prezent obiectul asociat cu cel de-al treilea bit (începând de la bitul cel mai semnificativ). Va trebui să executăm un AND între octetul respectiv şi masca 00100000, care va produce ca rezultat un octet cu toți biții 0, dacă şi numai dacă cel de-al treilea bit din bitmap are valoarea 0. Astfel, un program poate să decidă cum să continue introducând operația AND drept condiție într-o instrucțiune de ramificare. Pe de altă parte, dacă al treilea bit din bitmap ester 1 şi dorim să-l modificam la valoarea 0 fără a afecta ceilalți biți, va trebui să efectuăm operația AND între bitmap şi masca 11011111 şi apoi să stocăm rezultatul la valoarea inițială.

În timp ce operația AND servește la copierea unei părți a unui şir de biți plasându-se 0 în partea neduplicată, operația OR poate fi utilizată pentru a copia o parte a unui şir de biți plasându-se 1 pe pozițiile neduplicate. Pentru a face acest lucru , vom folosi de asemenea o mască, dar de aceasta dată vom pune 0 pe poziția biților care trebuie duplicați şi vom utiliza 1 pentru a indica pozițiile care nu vor fi duplicate. De exemplu, operația OR între un octet oarecare si 11110000 va produce un rezultat în care primii 4 biți vor fi 1 ,iar ceilalți 4 biți ai celuilalt operand, după cum se poate observa şi din următorul exemplu:

```
      11110000
OR    10101010
      11111010
```

În consecință, masca 11011111 poate fi utilizată într-o operație AND pentru a forța scrierea unui 0 pe poziția celui de-al treilea bit dintr-un şir de 8, iar masca 00100000 poate fi folosită într-o operație OR pentru a forța scrierea unui 1 pe aceeaşi poziție.

Una dintre utilizările principale ale operației XOR o reprezintă compunerea complementului unui şir de biți. De exemplu, observați relația dintre al doilea operand si rezultatul operației XOR prezentate in continuare:

```
      11111111
XOR   10101010
      01010101
```

Efectuarea unui XOR între un octet oarecare şi un octet cu toți biții 1 produce ca rezultat complementul primului octet.

2.5.2 Operații de rotire şi deplasare la nivel de bit

Operațiile din grupul operațiilor de rotire şi deplasare furnizează o posibilitate de deplasare a biților dintr-un registru şi sunt folosite adesea pentru rezolvarea problemelor de aliniere, cum

ar fi pregătirea unui octet pentru utilizarea într-o operație ulterioară de mascare sau manipularea mantisei reprezentărilor în virgulă mobilă. Aceste operații sunt clasificate după direcția de mişcare (dreapta sau stânga), ținându-se cont şi dacă procesul este circular. În cadrul regulilor de clasificare apar numeroase variante care conduc la terminologii mixte. Să studiem mai amănunțit noțiunile implicate.

Dacă vom considera pentru început un octet şi-i vom deplasa conținutul la dreapta sau la stânga, ne putem imagina că bitul de la capătul spre care se face deplasarea dispare, în timp ce la celălalt capăt al octetului apare un spațiu liber. Operațiile de deplasare se diferențiază între ele tocmai prin ceea ce se întâmpla cu acest bit suplimentar în timpul deplasării. Una dintre tehnici este să se plaseze bitul suplimentar în spațiul liber de la celălalt capăt. Rezultatul este o deplasare circulară, denumită rotație. Astfel dacă vom efectua de 8 ori o deplasare circulară la dreapta a unui octet vom obține ca rezultat acelaşi octet, iar şapte deplasări circulare la dreapta echivalează cu o singură deplasare circulară spre stânga.

Altă tehnică de deplasare elimină bitul de la capătul spre care se face deplasarea si completează cu 0 spațiul liber apărut la celălalt capăt. Ca nume pentru aceasta operație se utilizează adesea termenul de deplasare logică (logical shift). Asemenea deplasări la stânga pot fi utilizate pentru realizarea înmulțirii cu 2 a reprezentărilor în complement faţa de 2. De fapt deplasarea cifrelor binare la stânga corespunde înmulțirii cu 2, analog deplasării cifrelor zecimale în cazul înmulțirii cu 10. Similar, împărțirea la 2 se poate efectua prin deplasarea numărului binar la dreapta. În cazul efectuării de operații de deplasare, trebuie avut în vedere să nu se altereze bitul de semn utilizat în anumite sisteme de reprezentare. De aceea, vom întâlni adesea deplasări la dreapta care completează întotdeauna spațiul liber (ce apare pe poziția bitului de semn) cu valoarea inițială a bitului de semn. Deplasările care lasă bitul de semn nemodificat se numesc deplasări aritmetice (arithmetic shifts).

2.5.3 Operații aritmetice

Cu toate că am prezentat deja operațiile aritmetice de adunare, scădere, înmulțire şi împărțire, mai trebuie să facem câteva precizări. Mai întâi, după cum am arătat, aceste operații pot fi adesea efectuate utilizând doar operația de adunare, alături de negarea logică. Din acest motiv, din proiectare, unele calculatoare simple dispun numai de instrucțiuni de adunare sau eventual de adunare şi scădere.

Trebuie să menționăm de asemenea că pentru fiecare operație aritmetică există numeroase variante. Ne-am referit deja la acest lucru atunci când am prezentat operațiile de adunare disponibile pentru calculatorul prezentat Anexa C. De exemplu, în cazul adunării, dacă valorile care trebuie adunate sunt stocate utilizându-se notația în complement faţa de 2, operația de adunare trebuie realizată direct ca o adunare în binar. Însă dacă operanzii sunt stocați utilizându-se notația în virgulă mobilă, pentru adunare trebuie mai întâi efectuata extragerea mantiselor operanzilor, deplasarea acestora la stânga sau la dreapta în funcție de valoarea câmpurilor exponenților, verificarea biților de semn, adunarea, propiu-zisă şi apoi convertirea rezultatului în virgulă mobilă. Puteți observa că deşi ambele operații sunt considerate ca fiind operații de adunare, activitatea desfăşurată de calculator diferă. Din punctul de vedere al calculatorului, între cele două operații nu există nici o similitudine.

2.6 Comunicația dintre calculatoare şi dispozitivele periferice

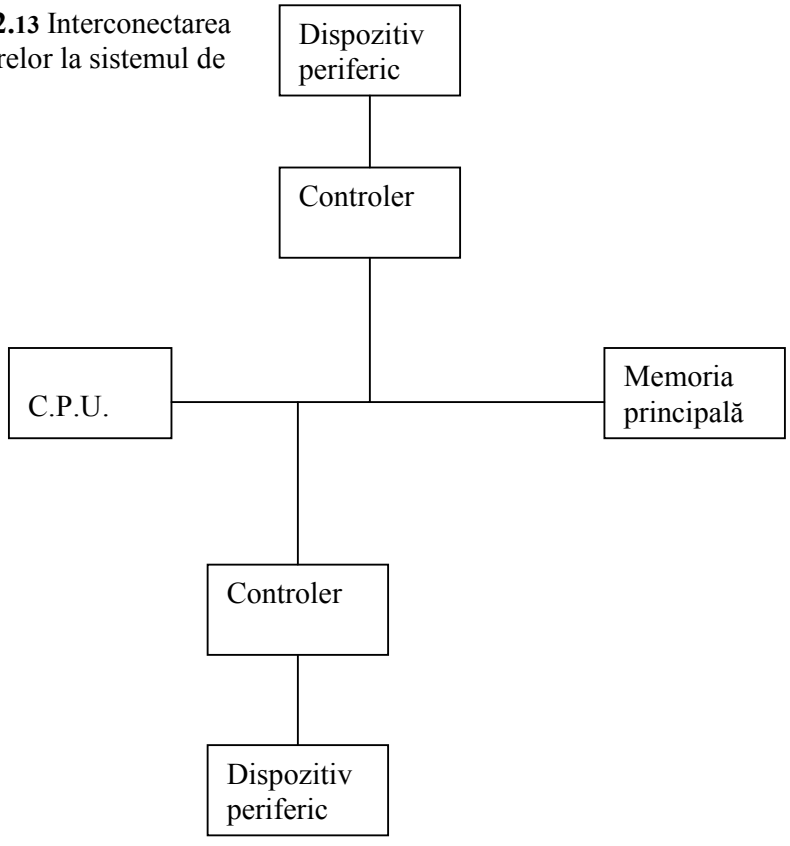
În acest subcapitol vom studia comunicația între unitatea centrală de prelucrare a unui calculator şi dispozitivele sale periferice, precum si rolul controlerelor în cadrul acestei comunicații.

2.6.1 Controlere

Comunicația dintre unitatea centrala de prelucrare a unui calculator şi un dispozitiv periferic oarecare este controlată de obicei de către un dispozitiv intermediar, denumit controler (controller). Fiecare controler gestionează comunicația cu un anumit tip de dispozitiv periferic. În cazul calculatoarelor personale, un controler corespunde din punct de vedere fizic unei plăci cu circuite electrice şi este adesea livrat împreună cu dispozitivul periferic pentru care este proiectat. Fiecare din aceste plăci este introdusă intr-un slot de pe placa de bază (placa mamă – motherboard) şi este conectată prin intermediul cablurilor electrice fie direct la dispozitivul periferic montat în calculator, fie la un conector aflat în partea din spate a calculatorului, conector la care se ataşează respectivul dispozitiv periferic.

Controlerul converteşte mesajele şi datele la forme compatibile cu caracteristicile interne ale calculatorului, respectiv la cele ale dispozitivului sau dispozitivelor periferice ataşate controlerului. Controlere sunt adesea ele însuşi mici calculatoare, fiecare având propriile circuite de memorie şi unități centrale de prelucrare care execută un program ce îi gestionează activitatea.

Figura 2.13 Interconectarea controlerelor la sistemul de calcul



Controlerele sunt ataşate la aceeaşi magistrală care conectează unitatea centrală de prelucrare a calculatorului la memoria principală (ca în figura 2.13). Fiecare controler monitorizează semnalele trimise de unitatea centrală de prelucrare a calculatorului şi răspunde atunci când observă că îi este adresat un semnal. Mai mult chiar, la intervale scurte de timp (de ordinul nanosecundelor) în unitatea centrală de prelucrare nu utilizează magistrala, controlerul poate scrie şi citi date din memoria principală a calculatorului. Abilitatea unui calculator de a accede la memoria principală a calculatorului poartă numele de acces direct la memorie (direct memory acces - DMA). Dacă controlerul unei unităţi de disc are acces direct la memorie, unitatea centrală de prelucrare poate trimite controlerului cereri codificate ca şiruri de biţi prin care să-i ceară să citească un anumit sector de pe disc şi să plaseze datele într-o anumită zonă precizată de celule de memorie. Apoi unitatea centrală de prelucrare poate continua execuţia altor operaţii în timp ce controlerul afectează operaţia de citire. Atunci când controlerul îşi termină de realizat sarcina atribuită, el trimite prin intermediul magistralei calculatorului un anumit semnal către unitatea centrală de prelucrare. Astfel de semnale speciale iau forma de întreruperi.

În mod similar, unitatea centrală de prelucrare poate să alcătuiască un bloc de date în memoria principală, apoi să ceară controlerului să copieze acel bloc pe un disc şi să-şi continue activitatea curentă până când controlerul semnalizează faptul că a terminat sarcina încredinţată.

Un bloc de memorie utilizat pentru transferul datelor spre şi dinspre dispozitivele periferice, ca în exemplele de mai sus, poartă numele de zonă de tampon (buffer). În general, numim bufer orice locaţie în care un sistem depune date pentru a fi prelucrate de alt sistem, acest proces fiind denumit lucrul cu buffere (buffering). Regiştrii dintr-o unitate centrală de prelucrare servesc ca zone tampon între unitatea de comandă şi unitatea aritmetico-logică, sau între unitatea centrală de prelucrare în ansamblu şi memoria principală.

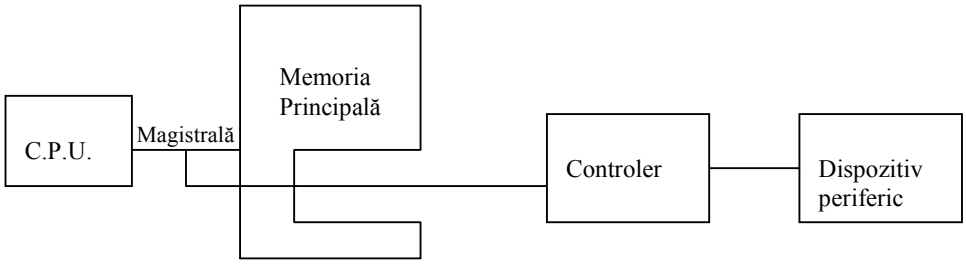
Ataşarea controlerelor la magistrala unui calculator măreşte semnificativ complexitatea operaţiilor de control al comunicaţiei de-a lungul acestei căi de comunicaţie. Trebuie efectuate transferuri de date între unitatea centrală de prelucrare şi memoria principală, între unitatea centrală de prelucrare şi fiecare controler, precum şi între fiecare controler şi memoria principală. Coordonarea acestor activităţi este un factor important în procesul proiectării calculatoarelor. Chiar şi în cazul unei proiectări foarte bune, magistrala principală poate deveni un punct critic, cunoscut sub numele de **gâtuirea von Neumann** (**von Neumann bottleneck**), datorită concurenţei pentru accesul la magistrală între unitatea centrală de prelucrare şi controlere.

2.6.2 Comunicaţia între unitatea centrală şi controlere

Comunicaţia între unitatea centrală de prelucrare a unui calculator şi a unui controler este controlată aproximativ la fel ca şi comunicaţia dintre unitatea centrală şi memoria principală. De fapt, în cazul multor calculatoare, controlerul este reprezentat de un bloc de celule din memoria principală. Atunci când unitatea centrală de prelucrare scrie un şir de biţi într-o celulă de memorie din cadrul acelui bloc de memorie, ca în cazul instrucţiunii STORE, şablonul este transferat de fapt controlerului şi nu memoriei. În mod similar, atunci când unitatea centrală de prelucrare încearcă să citească date dintr-una din aceste celule de memorie, ca de exemplu la instrucţiunea LOAD, ea primeşte un şir de biţi de la controler. Un

asemenea sistem de comunicație, denumit mapare în memorie a operațiilor de intrare /ieşire (memmory-mapped I/O), este reprezentat principial în figura 2.14.

Figura 2.14 Reprezentarea principială a mapării în memorie a operațiilor de intrare /ieşire



La calculatoarele care nu utilizează maparea în memorie a operațiilor de intrare /ieşire, fiecare controler are alocat un anumit grup de "adrese de intrare /ieşire". De exemplu, un controler ar putea avea alocate adresele de intrare /ieşire de la 10 la 1F (în notația hexazecimală), iar alt controler ar putea avea rezervate valorile de la 20 până la 2F. Aceste blocuri de adrese de intrare /ieşire joacă în cadrul mapării în memorie a operațiilor I/O acelaşi rol ca şi blocurile obişnuite de memorie, cu singura diferență că se utilizează doua sisteme de adresare diferite. Daca unitatea centrală de prelucrare scrie la adresa 12, datele vor fi memorate în memoria principală; dacă unitatea centrală de prelucrare scrie la adresa de intrare /ieşire 12, datele sunt prelucrate de un controller. La rândul lor, instrucțiunile în limbaj maşină pentru comunicația cu controlerele trebuie să fie diferite, deşi similare ca formă, de cele pentru comunicația cu memoria principală. De exemplu, în cazul calculatorului descris în Anexa C am rezervat codul de operație D pentru scrierea datelor la o adresă I/O. Mai precis, în timp ce instrucțiunea 383A transferă conținutul registrului 8 la adresa de memorie 2A, instrucțiunea D82A va transfera conținutul registrului 8 la adresa de intrare /ieşire 2A.

Indiferent dacă este sau nu folosită maparea în memorie a operațiilor de intrare /ieşire, blocul de adrese asociate unui controler este denumit **port,** el reprezentând "poartă " prin care informațiile intră şi ies din calculator. Fiecare adresă din cadrul unui port identifică o locație din controler în care datele pot fi depuse temporar. Aceste locații servesc drept zone tampon între calculator şi controler. Unele dintre ele sunt utilizate pentru stocarea datelor şi mesajelor care vin de la unitatea centrală de prelucrare; altele memorează şiruri de biți produşi de controler în scopul de a fi citiți de către unitatea centrală de prelucrare. Astfel, este posibilă existența unei căi de comunicație în ambele sensuri între controler şi unitatea centrală de prelucrare.

În cadru acestui proces, controlerul are o sarcină dublă. În unele cazuri, comunicația constă din mesaje între unitatea centrala şi controlerul însuşi; în alte cazuri, controlerul serveşte ca intermediar între mediul intern al calculatorului şi un dispozitiv periferic. În acest ultim caz, sarcina controlerului este aceea de a converti datele şi mesajele în formatele compatibile cu caracteristicile interne ale calculatorului şi ale dispozitivului periferic.

De asemenea, între calculator şi dispozitivul periferic pe care-l controlează are loc o comunicație în ambele sensuri, chiar dacă aparent comunicația este în sens unic. De exemplu, un calculator poate produce şi transmite caractere către controlerul unei imprimante mult mai

rapid decât le poate tipări imprimanta. De aceea, dacă n-ar exista o cale de comunicaţie în ambele sensuri între calculator şi imprimantă, imprimanta ar rămâne mereu în urmă.

Sintagma de realizare a unui dialog de confirmare (handshaking) se referă la comunicaţia în dublu sens care loc între dispozitive pentru evitarea acestor probleme de coordonare. În cazul unei imprimante, dialogul de confirmare este realizat de obicei proiectându-se imprimanta astfel încât să poată nu numai să recepţioneze date de la controler, ci să-şi transmită starea către acesta (de exemplu, sub numele de cuvânt de stare). În funcţie de sistem, controlerul poate să trateze el însuşi această informaţie de stare sau poate să o comunice unităţii centrale de prelucrare prin intermediul portului. În ambele cazuri, fie programul din controler, fie programul executat din unitatea centrală de prelucrare poate fi proiectat astfel încât fluxul de date de la imprimantă să fie întârziat până când se recepţionează de la aceasta informaţia de stare corespunzătoare.

2.6.3 Comunicaţie serială şi paralelă

Comunicaţia dintre diferite părţi ale unui sistem de calcul se efectuează intr-una dintre cele două forme elementare: serială şi paralelă. Aceşti termeni se referă la modul de transfer al şirurilor de biţi. În cazul comunicaţiei paralele (parallel communication), toţi biţii dintr-un sistem sunt transferaţi simultan, fiecare dintre ei pe o linie de separată. O asemenea tehnică permite transferul rapid al datelor, dar necesită o linie de comunicaţie relativ complicată, datorită utilizării unui număr mare de cabluri electrice. În schimb, comunicaţia serială (serial communication) se bazează pe transmiterea şirului bit cu bit . Această metodă este mai lentă dar poate fi realizată printr-o linie de comunicaţie mai simplă, deoarece biţii sunt transferaţi pe aceeaşi linie, unul după altul.

Calea mai simplă, suficientă pentru comunicaţia serială, permite calculatoarelor să transfere date utilizând sisteme de comunicaţie dezvoltate iniţial în alte scopuri. Un exemplu obişnuit îl constituie liniile telefonice, informaţiile digitale fiind convertite în semnale audio cu ajutorul unui dispozitiv numit modem (prescurtare de la modulator -demodulator), transmise serial de-a lungul liniilor telefonice şi apoi convertite în forma digitală la recepţie. Datorită limitărilor impuse de caracteristicile sistemului telefonic existent, o astfel de comunicaţie nu se poate realiza prin tehnicile de comunicaţie paralelă.

Viteza comunicaţiei seriale se măsoară în biţi pe secundă (bps), iar domeniul de variaţie se situează între câteva sute de biţi pe secundă şi milioane de biţi pe secundă. O altă unitate uzuală (folosită adesea incorect) este rata baud (baud rate), care se referă la viteza cu care se schimbă starea liniei pe care are loc comunicaţia. Pentru a clarifica această mărime, vom recurge la un exemplu. Dacă aplicăm un ton la unul dintre capetele unei linii telefonice, tonul poate fi detectat la capătul celălalt al liniei. În concluzie putem transmite mesaje prin intermediul liniei telefonice convenind că o anumită notă reprezintă valoarea 0, iar alta reprezintă valoarea 1.Într-un asemenea sistem linia de comunicaţie se poate afla în una dintre cele două stări: transmiterea unei note respectiv a celeilalte. De vreme ce fiecare stare reprezintă un singur bit, rata de transmisie a stărilor este identică cu rata de transmisie a biţilor, astfel că rata baud este egală cu rata de biţi pe secundă. Dar dacă modificăm protocolul de comunicaţie pentru a permite patru stări posibile ale liniei (patru note), fiecare stare poate reprezenta doi biţi. De exemplu, putem conveni să utilizăm o notă mai joasă pentru a reprezenta biţii 00, una ceva mai înaltă pentru biţii 01, una mai înaltă pentru biţii 10 şi o notă şi mai înaltă pentru biţii 11. În acest sistem, rata de transfer a biţilor este de două ori mai mare

decât rata cu care sunt transferate stările (notele), deoarece fiecare stare reprezintă doi biți. De aceea, rata de biți este de două ori mai mare decât rata baud.

În realitate, tehnica simplă de reprezentare a diferitelor modele de biți prin intermediul unor tonuri de diferite frecvențe (cunoscută ca modulație cu cheie de frecvență –frequencz-shift keying) este folosită numai pentru comunicații de viteză redusă. Pentru realizarea de modemuri cu rate de transfer de 2400 bps, 9600 bps şi mai mari, diferitele stări utilizate pentru reprezentarea şirurilor de biți sunt create prin combinarea modificărilor în frecvență, amplitudinea (volum) şi faza tonurilor (gradul de întârziere al tonului transmis). Ideea generală rămâne însă aceeaşi: prin reprezentarea mai multor biți cu ajutorul unei anumite stări, viteza de transfer a biților (măsurată în biți pe secundă) poate fi mult mai mare decât rata de variație a stărilor liniei (rata baud).

O altă metodă de creştere a eficienței transferului de date (precum şi a stocărilor datelor), compresia de date (data compression), reduce numărul de biți necesari pentru reprezentarea informațiilor. Sunt disponibile numeroase tehnici de compresie, fiecare dintre ele având atât avantaje cât şi dezavantaje. În cazul digitizării unei imagini ca şir de pixeli, s-ar putea să întâlnim şiruri lungi de pixeli identici. În această situație, adesea este mai eficient să codificăm culoarea comună a acestor pixeli şi lungimea şirului, în loc să descriem explicit fiecare pixel. Atunci când codificăm o secvență de cadre dintr-un film, adesea este mai avantajos să codificăm numai diferențele dintre două imagini consecutive în loc să codificăm în întregime fiecare cadru. Această metodă este cunoscută sub numele de codificare relativă (relative encodding).

În cazul reprezentărilor unor şiruri de caractere, se poate recurge la un cod Huffman (cunoscut de asemenea sub numele de cod dependent de frecvență). Un asemenea cod este proiectat astfel încât lungimea unui şir de biți care reprezintă un caracter să fie invers proporțională cu frecvența de utilizare a caracterului. Astfel, cele mai frecvent folosite caractere (în limba engleza, literele e, t, a şi i) sunt reprezentate prin şiruri scurte de biți, în timp ce literele mari rar utilizate (z, q, şi x) sunt reprezentate prin şiruri mai lungi. Se obține astfel o reprezentare mai scurtă a textului decât în cazul în care am utiliza un cod cu lungime uniformă, ca de exemplu codul ASCII.

Codificarea Lempel-Ziv constituie altă tehnică de compresie, aplicată atunci când codificarea datelor care reprezintă textul se face astfel încât să se înlocuiască secvențele care apar din nou cu o referire la ocurența precedentă, în loc să se repete secvența respectivă. În cazul unui text scris în limba engleză, această metodă poate fi utilizată pentru reducerea spațiului necesar copiilor repetate ale unor forme des întâlnite, ca ing sau the.

Eforturile de standardizare a tehnicilor de compresie a datelor au dus la includerea acestora în multe din modemurile existente în prezent pe piață. Atunci când două modemuri care utilizează scheme de compresie compatibile comunică între ele, modemul emițător comprimă datele înainte de a efectua transmisia, iar modemul receptor decomprimă datele după recepționarea lor. Folosind asemenea soluții, modemurile pot obține rate de transfer echivalente cu 56700 bps, chiar dacă de fapt sunt transmişi numai 14400 biți pe secundă, la o rată baud de 1200.