**Curs 3 şi 4**

Contents

[1. Organizarea unui sistem de calcul 3](#_Toc18578658)

[1. Organisation eines Computersystem 3](#_Toc18578659)

[2. Unitatea centrală (*Central Processing Unit* – CPU) 8](#_Toc18578660)

[2. Zentraleinheit (*Central Processing Unit* – CPU) 8](#_Toc18578661)

[2.1 Ceasul sistem 10](#_Toc18578662)

[2.1 Systemuhr 10](#_Toc18578663)

[2.2 Performanţele unui sistem de calcul 14](#_Toc18578664)

[2.2 Die Leistung eines Computersystems 14](#_Toc18578665)

[2.2.1 Viteza unui sistem de calcul 14](#_Toc18578666)

[2.2.1 Die Geschwindigkeit eines Computersystems 14](#_Toc18578667)

[2.2.2 Criterii de evaluare a performanţelor 15](#_Toc18578668)

[2.2.2 Leistungsbewertungskriterien 15](#_Toc18578669)

[2.3 „Dimensiunea” unui microprocesor 16](#_Toc18578670)

[2.3 Die „Größe“ eines Mikroprozessors 16](#_Toc18578671)

[3. Memoria 16](#_Toc18578672)

[3. Der Speicher 16](#_Toc18578673)

[3.1 Memoria internă 19](#_Toc18578674)

[3.1 Interner Speicher 19](#_Toc18578675)

[3.2 Memoria externă / secundară 22](#_Toc18578676)

[3.2 Externer / sekundärer Speicher 22](#_Toc18578677)

[3.3 Ierarhia memoriei 25](#_Toc18578678)

[3.3 Die Speicherhierarchie 25](#_Toc18578679)

[3.3.1. Memoria cache 26](#_Toc18578680)

[3.3.1 Der Cache-Speicher 26](#_Toc18578681)

[4. Dispozitivele periferice 29](#_Toc18578682)

[4. Die Peripheriegeräte 29](#_Toc18578683)

[4.1 Magistrale – structuri de interconectare 30](#_Toc18578684)

[4.1 Busse - Verbindungsstrukturen 30](#_Toc18578685)

[4.2. Magistrale I/O interne şi interfeţele asociate 33](#_Toc18578686)

[4.2. Interne E / A-Busse und die zugehörigen Schnittstellen 33](#_Toc18578687)

[4.2.1 Magistrala ISA 33](#_Toc18578688)

[4.2.1 ISA-Bus 33](#_Toc18578689)

[4.2.2 Magistrala PCI (*Peripheral Component Interconnect*) 34](#_Toc18578690)

[4.2.3 Magistrala AGP 35](#_Toc18578691)

[4.2.3 Der AGP Bus 35](#_Toc18578692)

[4.2.4 Magistrala PCI Express 35](#_Toc18578693)

[4.3 Magistrale I/O externe şi interfeţele asociate 39](#_Toc18578694)

[4.3 E/A Busse und zugehörige Schnittstellen 39](#_Toc18578695)

[4.3.1 IDE / ATA (*Integrated Drive Electronics / Advanced Technology Attachment*) 39](#_Toc18578696)

[4.3.1 IDE / ATA (*Integrated Drive Electronics / Advanced Technology Attachment*) 39](#_Toc18578697)

[4.3.2 SCSI (*Small Computer System Interface*) 40](#_Toc18578698)

[4.3.2 SCSI (*Small Computer System Interface*) 40](#_Toc18578699)

[4.3.3 Interfaţa serială 41](#_Toc18578700)

[4.3.3 Serielle Schnittstelle 41](#_Toc18578701)

[4.3.4 Interfaţa paralelă 41](#_Toc18578702)

[4.3.4 Parallele Schnittstelle 41](#_Toc18578703)

[4.3.5 USB (Universal Serial Bus) 42](#_Toc18578704)

[4.3.5 USB (Universal Serial Bus) 42](#_Toc18578705)

[5. Arhitectura microprocesoarelor x86 (IA-32) 42](#_Toc18578706)

[5. Architektur von x86-Mikroprozessoren (IA-32) 42](#_Toc18578707)

[5.1 Structura microprocesorului 42](#_Toc18578708)

[5.1 Aufbau des Mikroprozessors 42](#_Toc18578709)

[5.1.1 Regiştrii generali ai EU 43](#_Toc18578710)

[5.1.1 EU-Allgemeine Register 43](#_Toc18578711)

[5.1.2 Flagurile 45](#_Toc18578712)

[5.1.2 Flaggen 45](#_Toc18578713)

[5.1.3 Regiştrii de adresă şi calculul de adresă 47](#_Toc18578714)

[5.1.3 Adressregister und Adressberechnung 47](#_Toc18578715)

[5.2 Reprezentarea instrucţiunilor maşină 52](#_Toc18578716)

[5.2 Darstellung von MaschinenAnweisungen 52](#_Toc18578717)

[5.3 Adrese FAR şi NEAR 55](#_Toc18578718)

[5.3 FAR- und NEAR-Adressen 55](#_Toc18578719)

[5.4 Calculul *offset*-ului unui operand. Moduri de adresare 56](#_Toc18578720)

[5.4 Berechnung des *Offsets* eines Operanden. Adressierungsarten 56](#_Toc18578721)

|  |  |  |
| --- | --- | --- |
| 1. Organizarea unui sistem de calcul |  | 1. Organisation eines Computersystem |
| Un sistem de calcul (SC) este un dispozitiv care lucrează automat, sub controlul unui program memorat. El prelucrează date cu scopul de a produce nişte rezultate. Aceste rezultate se obţin ca efect al procesării.  Funcţiile de bază ale unui SC sunt:   * procesarea de date; * memorarea de date; * transferul de informaţii; * controlul tuturor componentelor SC.   Arhitectura unui sistem de calcul poate fi analizată la nivel:   * *structural* (componentele şi căile de comunicare între acestea); * *logic* (funcţiile fiecărei componente în cadrul structurii).   *Structura unui sistem de calcul este:*   * hardware - partea de echipamente:   + unitatea centrala de procesare (*Central Processing Unit* – CPU);   + memoria;   + dispozitivele periferice;   - software - partea de programe:   * + soft de sistem (aplicaţii destinate sistemului de calcul şi sistemului de operare);   + soft utilizator (restul aplicaţiilor);   - *firmware* - partea de microprograme.  Hardware-ul şi software-ul se prezintă sub forma unor niveluri (componente) ierarhice. Fiecare componentă inferioară ascunde detaliile de implementare faţă de componenta superioară imediată. O astfel de abordare este reflectarea **principiului abstractizării** sau al proiectării ierarhice.  Printre nivelurile ierarhice ale hardware-ului se pot identifica: structuri din siliciu sau alte materiale (componente electronice (tranzistori etc.), componente logice, circuite logice, unităţi funcţionale ale sistemului (ALU, CU etc.), componente ale calculatorului (CPU, memorie, sistem de I/O), iar nivelurile ierarhice ale software-ului sunt reprezentate de limbajul maşină, limbajul de asamblare şi limbajele de nivel înalt.  *Arhitectura (organizarea) unui sistem de calcul* se referă la acele atribute ale sistemului care sunt vizibile programatorului şi care au un impact direct asupra execuţiei unui program: setul de instrucţiuni maşină, caracteristicile de reprezentare a datelor, modurile de adresare şi sistemul de intrare / ieşire (I/O). Din punct de vedere organizatoric, componentele unui SC sunt:   * Unitatea de Control; * Unitatea de Execuţie, numită şi Calea de Date; * Memoria; * Sistemul de Intrare (*input*) / iEşire (*output*) = sistemul de I/O sau de I/E; * Structuri de interconectare a componentelor de mai sus (Magistrale),   unde Unitatea de Control şi Calea de date sunt componente ale procesorului.  Această organizare este independentă de tehnologia hardware adoptată pentru construcţia sistemului de calcul. Orice componentă a unui SC poate fi încadrată în una din aceste 5 categorii.  Văzută dintr-un alt unghi, arhitectura unui SC este compusă din mulţimea instrucţiunilor maşină şi organizarea maşinii.  Mulţimea instrucţiunilor maşină (*Instruction Set Architecture* – ISA) este o interfaţă cheie între nivelurile de abstractizare. Ea este interfaţa dintre hardware şi software-ul de nivel scăzut. O astfel de interfaţă permite unor implementări diferite ale SC să ruleze programe identice, caz în care vorbim despre calculatoare compatibile (de exemplu – „calculatoarele compatibile IBM PC” nu au acelaşi hardware, dar au acelaşi set de instrucţiuni).  ISA defineşte:   * organizarea SC, modul de stocare a informaţiei (registre, memorie); * tipurile şi structurile de date (codificări, reprezentări); * formatul instrucţiunilor; * setul de instrucţiuni (codurile operaţiilor) pe care microprocesorul le poate efectua; * modurile de adresare şi de accesare a datelor şi instrucţiunilor; * condiţiile de excepţie;   Organizarea unei maşini se referă la:   * implementarea, capacitatea şi performanţa unităţilor funcţionale; * interconexiunile dintre aceste unităţi; * fluxul de informaţie dintre unităţi; * controlul fluxului de informaţie.   Majoritatea calculatoarelor actuale sunt construite pe baza arhitecturii von Neumann. Ideea de plecare este de a utiliza memoria internă pentru a stoca secvenţe de control în vederea îndeplinirii unei anumite sarcini – secvenţe de programe; astfel, putem vorbi despre maşini programabile. Acesta reprezintă aşa-numitul concept al **programului memorat** (*stored-program concept*). În contrast, primele maşini erau construite pentru îndeplinirea unei anumite operaţii şi era necesară modificarea acestora pentru a putea efectua un alt tip de operaţie. |  | Ein Computersystem (CS) ist ein Gerät, das automatisch unter der Steuerung eines gespeicherten Programms arbeitet. Er verarbeitet Daten, um Ergebnisse zu erzielen. Diese Ergebnisse werden als Verarbeitungseffekt erhalten.  Die Grundfunktionen eines CS sind:   * Datenverarbeitung; * Datenspeicherung; * Übermittlung von Informationen; * Steuerung aller CS-Komponenten.   Die Architektur eines Computersystems kann auf folgender Ebene analysiert werden:   * *strukturell* (die Komponenten und Kommunikationswege zwischen ihnen); * *logisch* (die Funktionen jeder Komponente innerhalb der Struktur).   *Die Struktur eines Computersystems ist*:   * Hardware - Teil der Ausrüstung: * eine Zentraleinheit (*Central Processing Unit* – CPU); * ein Speicher; * Peripheriegeräte; * Software – der Teil von Programmen: * Systemsoftware (Anwendungen für das Computersystem und das Betriebssystem); * ein Software-Benutzer (der Rest der Anwendungen); * *firmware* – der Teil den Mikroprogrammen.   Die Hardware und Software besteht aus hierarchischen Ebenen (Komponenten). Jede untere Komponente verbirgt die Implementierungsdetails vor der unmittelbar oberen Komponente. Ein solcher Ansatz spiegelt **das Prinzip der Abstraktion** oder hierarchischen Gestaltung wider.  Unter den Hierarchieebenen der Hardware können wir identifizieren: Siliziumstrukturen oder andere Materialien (elektronische Komponenten (Transistoren usw.), Logikkomponenten, Logikschaltungen, Funktionseinheiten des Systems (ALU, CU usw.), Computerkomponenten (CPU, Speicher, E / A-System) und die Hierarchieebenen der Software werden durch Maschinensprache, Assemblersprache und übergeordnete Sprachen dargestellt.  *Die Architektur (Organisation) eines Computersystems* bezieht sich auf diejenigen Systemattribute, die für den Programmierer sichtbar sind und die direkten Einfluss auf die Ausführung eines Programms haben: den Maschinenbefehlssatz, die Datenrepräsentationseigenschaften, die Adressierungsmodi und das Eingabe- / Ausgabesystem. (I / O). Aus organisatorischer Sicht sind die Komponenten eines CS:   * Die Steuereinheit; * Die Ausführungseinheit, auch Datenpfad genannt; * Der Speicher; * Der Eingabe- / Ausgabesystem = der E / A-System; * Verbindungsstrukturen der oben genannten Komponenten (Magistrale),   dabei sind die Steuereinheit und der Datenpfad Komponenten des Prozessors.  Diese Organisation ist unabhängig von der für den Aufbau des Computersystems verwendeten Hardwaretechnologie. Jede Komponente eines CS kann in eine dieser 5 Kategorien fallen.  Aus einem anderen Blickwinkel besteht die Architektur eines SC aus der Vielzahl von Maschinenbefehlen und der Organisation der Maschine.  Die Befehlssatzarchitektur (*Instruction Set Architecture,* ISA) ist eine wichtige Schnittstelle zwischen Abstraktionsebenen. Es ist die Schnittstelle zwischen niedriger Pegel Hardware und Software. Eine solche Schnittstelle ermöglicht es verschiedenen CS-Implementierungen, identische Programme auszuführen. In diesem Fall handelt es sich um kompatible Computer (z. B. "IBM PC-kompatible Computer" haben nicht dieselbe Hardware, aber denselben Befehlssatz).  ISA definiert:  - Organisation von der CS, Informationsspeichermodus (Register, Speicher);  - Datentypen und -strukturen (Kodierung, Darstellungen);  - Format der Anweisungen;  - den Befehlssatz (die Operationscoden), den der Mikroprozessor ausführen kann;  - Möglichkeiten der Adressierung und des Zugriffs auf Daten und Anweisungen;  - außergewöhnliche Bedingungen;  Die Organisation eines CS bezieht sich auf:  - Implementierung, Kapazität und Leistung von Funktionseinheiten;  - die Verbindungen zwischen diesen Einheiten;  - Informationsfluss zwischen Einheiten;  - Kontrolle des Informationsflusses.  Die meisten aktuellen Computer basieren auf der von Neumann-Architektur. Die Idee des Startens besteht darin, den internen Speicher zum Speichern von Steuersequenzen zu verwenden, um eine bestimmte Aufgabe auszuführen - Programmsequenzen; Somit können wir über programmierbare Maschinen sprechen. Dies ist das sogenannte **Speicherprogrammkonzept** (*stored-program concept*). Im Gegensatz dazu wurden die ersten Maschinen gebaut, um eine bestimmte Operation auszuführen, und es war notwendig, sie zu modifizieren, um eine andere Art von Operation auszuführen. |
| Figura (*Abbildung*) 1. *Arhitectura unui sistem de calcul (*Die Architektur eines Computersystems*)* | | |
| Caracteristicile arhitecturii von Neumann:   * atât datele, cât şi instrucţiunile sunt reprezentate ca şiruri de biţi şi sunt stocate într-o memorie *read-write*. Nu se poate face diferenţa între date şi instrucţiuni prin simpla citire a unei locaţii de memorie – trebuie cunoscut ce reprezintă pentru a i se stabili semnificaţia (de exemplu: valoarea 98h - poate reprezenta o valoare a unei variabile de dimensiune egală cu un cuvânt sau codul instrucţiunii maşină corespunzător instrucţiunii „8086 CBW” din limbajul de asamblare); * conţinutul memoriei se poate accesa în funcţie de locaţie (adresă), indiferent de tipul informaţiei conţinute; * execuţia unui set de instrucţiuni se efectuează secvenţial, prin citirea de instrucţiuni consecutive din memorie.   O altă perspectivă este oferită de Figura 2, în care se pot vedea etapele execuţiei unui program într-un sistem de calcul construit pe baza paradigmei von Neumann. Arhitectura este reprezentată într-o formă simplificată. |  | Merkmale der von Neumann-Architektur:  - Sowohl Daten als auch Anweisungen werden als Zeichenfolgen von Bits dargestellt und in einem Lese- / Schreibspeicher gespeichert. Es ist nicht möglich, zwischen Daten und Anweisungen durch einfaches Lesen eines Speicherorts zu unterscheiden – es muss bekannt sein, was es darstellt, um seine Bedeutung zu bestimmen (zum Beispiel: der Wert 98h – es kann einen Wert einer Variablen darstellen, die die gleiche Größe wie ein Wort oder der Maschinenanweisungscode hat, der der Anweisung "8086 CBW" in der Assemblersprache entspricht);  - Auf den Inhalt des Speichers kann in Abhängigkeit von der Position (Adresse) unabhängig von der Art der enthaltenen Informationen zugegriffen werden.  - Die Ausführung eines Befehlssatzes wird sequentiell ausgeführt, indem aufeinanderfolgende Anweisungen aus dem Speicher gelesen werden.  Eine andere Perspektive bietet Abbildung 2, in der man die Schritte der Ausführung eines Programms in einem Computersystem nach dem von Neumann-Muster sehen können. Die Architektur ist vereinfacht dargestellt. |
| **Date intrare**  **Date**  **ieşire**  **Memoria**  **CPU**  Program  **a)**  **Date intrare**  **Date**  **ieşire**  **Memoria**  **CPU**  Program  **b)**  **Date intrare**  **Date**  **ieşire**  **Memoria**  **CPU**  Instrucţiune  **c)**  **Date intrare**  **Date**  **ieşire**  **Memoria**  **CPU**  Program  **d)**  Figura 2. *Etapele execuţiei unui program (*Die Phasen der Ausführung eines Programms*)* | | |
| 2. Unitatea centrală (*Central Processing Unit* – CPU) |  | 2. Zentraleinheit (*Central Processing Unit* – CPU) |
| Procesorul controlează modul de operare a calculatorului şi execută funcţiile de procesare a datelor.  Funcţiile unui CPU sunt:   * obţinerea instrucţiunilor care trebuie executate; * obţinerea datelor necesare instrucţiunilor; * procesarea datelor (execuţia instrucţiunilor); * furnizarea rezultatelor obţinute.   Putem identifica două componente de bază la nivelul unui CPU:   * Unitatea Aritmetică-Logică (*Arithmetic Logic Unit –* ALU); * Unitatea de Comandă şi Control (*Control Unit –* CU);   iar pentru îndeplinirea funcţiilor de mai sus, CPU mai are nevoie de:   * registre – acestea sunt dispozitive de stocare temporară a datelor şi informaţiilor de control (instrucţiunile), de capacitate mică şi viteză de acces mare; * magistrale interne CPU – dispozitive pentru comunicare între componentele CPU şi comunicare cu exteriorul.   *Unitatea Aritmetică-Logică* execută operaţii aritmetice şi logice asupra datelor.  *Unitatea de Comandă şi Control* este componenta CPU care dirijează toate componentele sistemului de calcul prin controlul direct asupra acestora. Pentru aceasta, trimite semnale de control în interiorul CPU şi către magistrala sistem, sau captează semnale dinspre magistrala sistem. Nu execută instrucţiuni, ci le decodifică şi comandă altor componente execuţia efectiva a acestora. Regulile după care funcţionează CU sunt codificate în *Programmable Logic Array* (PLA – dispozitiv programabil utilizat pentru implementarea de circuite logice combinaţionale AND / OR) sau într-o memorie de tip ROM (*Read-Only Memory*).  Pentru execuţia unei instrucţiuni, aceasta şi datele necesare trebuie aduse din memoria secundară sau de la *un dispozitiv de intrare* în memoria principală. CU coordonează *ciclul de execuţie a* *instrucţiunii:*   * citirea instrucţiunii din memoria principală (FETCH); * decodificarea acesteia (DECODE *-* tipul instrucţiunii, numărul de argumente etc.); * obţinerea operanzilor necesari instrucţiunii (READ MEMORY); * execuţia ei (EXECUTE; dacă e cazul, ALU primeşte controlul pentru efectuarea operaţiei aritmeticesau logice implicate); * furnizarea rezultatelor în registre sau în memorie (STORE).   Apoi, CU poateiniţia transferul acestor date rezultate din memoria internă către un dispozitiv de ieşiresau către un dispozitiv de stocare secundar.  Cunoscând structura unui *CPU* şi paşii din ciclul de execuţie a unei instrucţiuni, putem contura structurafuncţională a procesorului: *modulul de control* şi *calea de date (Datapath).* |  | Der Prozessor steuert die Funktionsweise des Computers und führt Datenverarbeitungsfunktionen aus.  Die Funktionen einer CPU sind:   * Einholen der auszuführenden Anweisungen; * Einholen der für die Anweisungen erforderlichen Daten; * Datenverarbeitung (Ausführung von Anweisungen); * Bereitstellung der erzielten Ergebnisse.   Wir können zwei grundlegende Komponenten einer CPU identifizieren:  - Arithmetisch-Logische Einheit (*Arithmetic Logic Unit* – ALU);  - Befehls- und Steuereinheit (*Control Unit* – CU);  und um die obigen Funktionen auszuführen, benötigt die CPU:  - Registeren – Dies sind Geräte zur vorübergehenden Speicherung von Steuerdaten und -informationen (Anweisungen) mit geringer Kapazität und hoher Zugriffsgeschwindigkeit.  - CPU-interne Busse – Geräte für die Kommunikation zwischen CPU-Komponenten und die Kommunikation nach außen.  Die *Arithmetic-Logic-Einheit* führt arithmetische und logische Operationen an den Daten aus.  Die Steuer- und Befehlseinheit ist die CPU-Komponente, die alle Komponenten des Computersystems durch direkte Steuerung steuert. Dazu sendet es Steuersignale innerhalb der CPU und an den Systembus oder empfängt Signale vom Systembus. Es führt keine Anweisungen aus, sondern decodiert und befiehlt anderen Komponenten, diese effektiv auszuführen. Die Regeln, nach denen CU arbeitet, sind im *Programmable Logic Array* (PLA – ein programmierbares Gerät, das für die Implementierung von UND / ODER-Verknüpfungslogikschaltungen verwendet wird) oder in einem ROM (Nur-Lese-Speicher, *Read-Only Memory*) codiert.  Zur Ausführung einer Anweisung müssen dieser und die erforderlichen Daten aus dem Sekundärspeicher oder von einem *Eingabegerät* in den Hauptspeicher gebracht werden. Der CU koordiniert den Ausführungszyklus der Anweisung:   * Lesen die Anweisung aus dem Hauptspeicher (FETCH); * Decodierung (DECODE – Art der Anweisung, Anzahl der Argumente usw.); * Erhalten der für den Befehl benötigten Operanden (READ MEMORY); * seine Ausführung (EXECUTE; ALU erhält gegebenenfalls die Steuerung zur Durchführung der betreffenden arithmetischen oder logischen Operation); * Bereitstellen von Ergebnissen in Registern oder im Speicher (STORE).   Dann kann die CU die Übertragung dieser Daten vom internen Speicher zu einem Ausgabegerät oder einem sekundären Speichergerät initiieren.  Wenn wir die Struktur einer CPU und die Schritte im Ausführungszyklus einer Anweisung kennen, können wir die Funktionsstruktur des Prozessors skizzieren: *das Steuermodul* und *den Datenpfad* (*Datapath*). |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | 1 | 2 | 3 | 4 | 5 | | **FETCH** | **DECODE** | **READ MEMORY** | **EXECUTE** | **STORE sau (*oder*)**  **WRITE BACK** | | Adresa instrucţiunii (*Adresse der Anweisung*)  Copierea  instrucţiunii într-un registru (*Kopieren die*  *Anweisung in einem Register*) | Decodificare instrucţiune (*Dekodierung der Anweisung*)  Generare semnale pentru execuţie (*Signale zur Ausführung generieren*) | Citire operanzi din memorie (*Liest Operanden aus dem Speicher*) | Efectuare operaţie (UAL)  (*Operationsausführung* - *ALU*) | Scriere rezultat (*Schreiben das Ergebnis*)  Scriere stare (*Schreibstatus*) | |  |  | *Uneori aceste 2 faze sunt prezentate combinate în una singură, EXECUTE*  (Manchmal werden diese beiden Phasen zu einer zusammengefasst) | |  |   Figura 3. *Etapele execuţiei unui instrucţiuni (*Die Schritte zum Ausführen einer Anweisung*)* | | |
| **Modulul de control** este răspunzător de comunicarea cu exteriorul CPU (preluarea şi interpretarea instrucţiunilor şi transmiterea rezultatelor), precum şi de controlul execuţiei instrucţiunilor (emiterea semnalelor de control către calea de date şi recepţionarea semnalelor de stare dinspre aceasta).  Din **Calea de date** fac parte componente de stocare (registre de date), unităţi funcţionale (ALU) şi căi de comunicare. Practic, calea de date defineşte mulţimea operaţiilor efectuate asupra datelor în vederea execuţiei unei instrucţiuni şi drumul parcurs de datele prelucrate în decursul execuţiei operaţiilor. |  | **Das Steuermodul** ist verantwortlich für die Kommunikation mit der externen des CPU (Entgegennahme und Interpretation der Anweisungen und Übertragung der Ergebnisse) sowie für die Steuerung der Ausführung der Anweisungen (Senden der Steuersignale an den Datenpfad und Empfangen der Statussignale von dieser).  **Der Datenpfad** enthält Speicherkomponenten (Datenregister), Funktionseinheiten (ALU) und Kommunikationspfade. Grundsätzlich definiert der Datenpfad den Satz von Operationen, die an den Daten ausgeführt werden, um eine Anweisung auszuführen, und den Pfad, den die während der Ausführung der Operationen verarbeiteten Daten zurücklegen. |
| 2.1 Ceasul sistem |  | 2.1 Systemuhr |
| Fiecare CPU are un ceas intern care produce şi trimite semnale electrice pe magistrala de control pentru a sincroniza operaţiile sistemului. Semnalele alternează valori 0 şi 1 cu o anumită frecvenţă numită *frecvenţa* ceasului sistem. Timpul necesar trecerii din starea 0 în 1 şi apoi iar în 0 se numeşte perioada ceasului sau ciclu de ceas (*clock cycle*).  Frecvenţa de ceas este de fapt numărul de cicluri de ceas pe secundă şi este măsurată în hertzi.  Observaţie: 1 Hz = 1 ciclu de ceas / secundă. Frecvenţa = 1 / perioada |  | Jede CPU verfügt über eine interne Uhr, die elektrische Signale erzeugt und an den Steuerbus sendet, um den Systembetrieb zu synchronisieren. Die Signale wechseln sich mit 0 und 1 mit einer bestimmten Frequenz ab, die als Systemtaktfrequenz bezeichnet wird. Die Zeit, die benötigt wird, um vom Zustand 0 auf 1 und dann wieder auf 0 zu schalten, wird als Taktperiode oder Taktzyklus (*clock cycle*) bezeichnet.  Die Taktfrequenz ist tatsächlich die Anzahl der Taktzyklen pro Sekunde und wird in Hertz gemessen.  Hinweis: 1 Hz = 1 Taktzyklus / Sekunde. Frequenz = 1 / Periode |
| Timp (*zeit*)  Un ciclu (*ein Zyklus*)  1  0  Figura 4. *Semnalul periodic de tact sau* clock (Periodisches Takt- oder clock Signal) | | |
| Toate operaţiile efectuate de microprocesor sunt sincronizate cu ceasul sistem; aceasta înseamnă că procesorul nu poate efectua de o manieră secvenţială operaţii mai rapid decât frecvenţa de tact a ceasului (a unităţii de timp a ceasului). Astfel, un procesor cu o viteză de ceas de 200 MHz are un ceas intern care generează 200.000.000 de impulsuri pe secundă. Un ciclu de ceas pentru un astfel de procesor are o durată de 1 / 200.000.000 secunde.  Un ciclu de ceas este în general, pentru un procesor, unitatea de bază pentru măsurarea timpului. Este cea mai mică cuantă de timp sesizabilă de către procesor. Toate activităţile şi instrucţiunile executate de procesor sunt executate în general în multipli ai ciclului de ceas (din punct de vedere al duratei de execuţie). Fiecare instrucţiune maşină necesită un număr fix, cunoscut (în general) de cicluri procesor. Există diferenţe în durata execuţiei instrucţiunilor maşină în funcţie de natura parametrilor şi de mediul lor de stocare. Astfel o instrucţiune **MOV** *destinaţie*, *sursă* se execută între 2 şi 14 cicluri de ceas procesor în funcţie de natura argumentelor destinaţie şi sursă care pot fi: regiştri generali, regiştri de segment, constante, adrese de memorie. Fiecare combinaţie posibilă a acestor tipuri de parametri va rezulta într-o durată de execuţie specifică în cicluri de ceas procesor.  Timpul de execuţie al unei instrucţiuni măsurat în cicluri de ceas procesor poartă denumirea *Cycles per Instruction* (*CPI*).Încă de la primele procesoare apărute pe piaţă, modul de execuţie al instrucţiunilor maşină a fost unul *secvenţial*. Procesorul preia o instrucţiune, petrece un număr de cicluri de ceas la execuţia completă a acesteia, după care trece la următoarea instrucţiune ş.a.m.d.  În general CPI se referă la numărul *mediu* de cicluri procesor per instrucţiune executată de procesor – în raport cu toate instrucţiunile ce compun un program sau în raport cu întreg setul de instrucţiuni al procesorului.  Observaţie: Deşi o instrucţiune **MOV** se execută în acelaşi număr *N* de cicluri procesor, durata de execuţie în timp (secunde) este diferită de la un procesor la altul. Astfel un procesor ce rulează la viteza de 200 MHz va executa o instrucţiune **MOV** în 20 nanosecunde, în timp ce un procesor ce rulează la 1 GHz o va executa de aproximativ 5 ori mai rapid (4 nanosecunde). |  | Alle vom Mikroprozessor ausgeführten Operationen werden mit der Systemuhr synchronisiert. Dies bedeutet, dass der Prozessor keine Operationen ausführen kann, die sequentiell schneller als die Taktfrequenz (Takteinheit der Zeit) sind. Ein Prozessor mit einer Taktrate von 200 MHz verfügt somit über einen internen Takt, der 200.000.000 Impulse pro Sekunde erzeugt. Ein Taktzyklus für einen solchen Prozessor hat eine Dauer von 1 / 200.000.000 Sekunden.  Ein Taktzyklus ist für einen Prozessor in der Regel die Grundeinheit für die Zeitmessung. Dies ist die kürzeste Zeit, die der Prozessor erkennen kann. Alle vom Prozessor ausgeführten Aktivitäten und Anweisungen werden im Allgemeinen in Vielfachen des Taktzyklus (in Bezug auf die Ausführungszeit) ausgeführt. Jeder Maschinenbefehl erfordert eine feste Nummer, die (allgemein) aus den Prozessorzyklen bekannt ist. Die Ausführung der Maschinenbefehle unterscheidet sich je nach Art der Parameter und deren Speicherumgebung. Somit läuft eine **MOV** *Ziel, Quelle* Anweisung, zwischen 2 und 14 Prozessortaktzyklen, abhängig von der Art der Ziel- und Quellargumente, die sein können: allgemeine Register, Segmentregister, Konstanten, Speicheradressen. Jede mögliche Kombination dieser Parametertypen führt zu einer bestimmten Ausführungszeit in Prozessortaktzyklen.  Die Ausführungszeit einer Anweisung, gemessen in Prozessortaktzyklen, wird als „*Cycles per Instruction*“ (CPI) bezeichnet. Seit die ersten Prozessoren auf dem Markt waren, wurden die Maschinenbefehle *sequentiell* ausgeführt. Der Prozessor nimmt einen Anweisung entgegen, verbringt bei seiner vollständigen Ausführung eine Anzahl von Taktzyklen und fährt dann mit dem nächsten Anweisung fort und so weiter.  Im Allgemeinen bezieht sich der CPI auf die *durchschnittliche* Anzahl von Prozessorzyklen pro Anweisung, die vom Prozessor ausgeführt werden – in Bezug auf alle Anweisungen, die ein Programm bilden, oder in Bezug auf den gesamten Befehlssatz des Prozessors.  Hinweis: Obwohl eine **MOV** Anweisung mit der gleichen Anzahl von *N* Prozessorzyklen ausgeführt wird, unterscheidet sich die Ausführungszeit (in Sekunden) von Prozessor zu Prozessor. Ein Prozessor, der mit einer Geschwindigkeit von 200 MHz läuft, führt einen **MOV** Anweisung mit 20 Nanosekunden aus, während ein Prozessor, der mit 1 GHz läuft, ihn etwa fünfmal schneller ausführt (4 Nanosekunden). |
| Întrucât în zilele noastre s-a ajuns la o limită tehnologică din punct de vedere al frecvenţei de ceas a procesoarelor, se urmăreşte creşterea vitezei acestora prin alte metode. Cea mai des întâlnită este *paralelizarea execuţiei instrucţiunilor maşină*, realizată la nivel de instrucţiune maşină. Ea se referă la paralelizarea etapelor execuţiei unei instrucţiuni, conform Figurii 5. Această tehnică de paralelizare se numeşte *pipelining* (de la cuvântul *pipeline* – conductă).  Prin această tehnică nu creşte viteza de execuţie a unei instrucţiuni (ea se execută în acelaşi număr de cicluri de ceas), ci se măreşte numărul de instrucţiuni executate de către procesor per unitate de timp. |  | Da heute eine technologische Grenze hinsichtlich der Taktfrequenz der Prozessoren erreicht ist, soll deren Geschwindigkeit durch andere Methoden gesteigert werden. Am gebräuchlichsten ist *die Parallelisierung der Ausführung der Maschinenbefehle* auf der Ebene der Maschinenbefehle. Es bezieht sich auf die Parallelisierung der Schritte zum Ausführen eines Befehls gemäß Abbildung 5. Diese Parallelisierungstechnik wird *Pipelining* (von der Wort *Pipeline*) genannt.  Diese Technik erhöht nicht die Ausführungsgeschwindigkeit einer Anweisung (sie wird in der gleichen Anzahl von Taktzyklen ausgeführt), sondern erhöht die Anzahl der vom Prozessor pro Zeiteinheit ausgeführten Anweisungen. |
| |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | | F I1 | D I1 | R I1 | E I1 | WB I1 |  |  |  |  | |  | F I2 | D I2 | R I2 | E I2 | WB I2 |  |  |  | |  |  | F I3 | D I3 | R I3 | E I3 | WB I3 |  |  | |  |  |  | F I4 | D I4 | R I4 | E I4 | WB I4 |  | |  |  |  |  | F I5 | D I5 | R I5 | E I5 | WB I5 | |  |  |  |  |  |  |  |  | Timp  (*Zeit*) |   Figura 5. *Mecanismul de paralelizare în execuţia instrucţiunilor* (pipelining) (Der Parallelisierungsmechanismus bei der Ausführung der Anweisungen) | | |
| Observaţii:   1. Ciclurile FETCH, DECODE, READ MEMORY, EXECUTE şi WRITE BACK (sau STORE) nu au durate egale, putând varia de la o arhitectură a SC la alta. 2. Numărul de instrucţiuni / ciclu de ceas – *Instructions per Cycle* (*IPC*) – este un termen care poate crea confuzie dacă ne gândim că CPU poate executa mai mult de o singură acţiune / instrucţiune în fiecare ciclu de ceas. Tehnica de pipelining permite însă execuţia *virtual paralelă* a mai multor instrucţiuni în acelaşi timp. Cu cât *pipeline*-ul este mai lung şi numărul de cicluri de ceas pentru instrucţiuni mai mic, cu atât creşte factorul IPC pentru procesorul respectiv. Pentru exemplul din Figura 6, dacă am considera că fiecare fază F, D, RM, E şi WB durează acelaşi număr de cicluri de tact (să zicem, 1 ciclu), atunci IPC = 5 / 9 = 0.55, iar CPI = 9 / 5 = 1.8. 3. Prelungind timpul la infinit şi trecând la limită, IPC va tinde la 1. 4. Dacă procesorul ar avea mai multe (în Figura 6, două) unităţi de execuţie şi ar prezice corect ce instrucţiuni se pot executa în paralel, ar putea ajunge ca IPC să fie supraunitar. Aceste SC se numesc superscalare. În Figura 6, IPC = 10 / 9 = 1.11. La ciclul 5 se execută simultan 2 instrucţiuni complete şi toate unităţile funcţionale ale procesorului sunt solicitate. |  | Bemerkungen:   1. Die Zyklen FETCH, DECODE, READ MEMORY, EXECUTE und WRITE BACK (oder STORE) haben nicht die gleiche Dauer und können von einer SC-Architektur zur anderen variieren. 2. Anzahl der Anweisungen / Taktzyklus – *Instructions per Cycle* (IPC) – ist ein Begriff, der verwirrend sein kann, wenn wir glauben, dass die CPU in jedem Taktzyklus mehr als eine Aktion / Anweisung ausführen kann. Die Pipelining-Technik ermöglicht jedoch die virtuelle parallele Ausführung mehrerer Befehle gleichzeitig. Je länger die *Pipeline* ist und je kürzer die Anzahl der Taktzyklen für Anweisungen ist, desto höher ist der IPC-Faktor für diesen Prozessor. Wenn wir für das Beispiel in Abbildung 6, berücksichtigen, dass jede Phase F, D, RM, E und WB dieselbe Anzahl von Taktzyklen (z. B. 1 Zyklus) dauert, ist IPC = 5/9 = 0,55 und CPI = 9/5 = 1,8. 3. Wenn Sie die Zeit bis zum Unendlichen verlängern und das Limit erreichen, tendiert der IPC zu 1. 4. Wenn der Prozessor mehr (in Abbildung 6 zwei) Ausführungseinheiten hat und korrekt voraussagt, welche Anweisungen parallel ausgeführt werden können, ist der IPC möglicherweise zu hoch. Diese CSs werden superskalar genannt. In Abbildung 6 ist der IPC = 10/9 = 1.11. In Zyklus 5 werden 2 vollständige Anweisungen gleichzeitig ausgeführt und alle Funktionseinheiten des Prozessors angefordert. |
| |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | | F I1 | D I1 | R I1 | E I1 | WB I1 |  |  |  |  | | F I2 | D I2 | R I2 | E I2 | WB I2 |  |  |  |  | |  | F I3 | D I3 | R I3 | E I3 | WB I3 |  |  |  | |  | F I4 | D I4 | R I4 | E I4 | WB I4 |  |  |  | |  |  | F I5 | D I5 | R I5 | E I5 | WB I5 |  |  | |  |  | F I6 | D I6 | R I6 | E I6 | WB I6 |  |  | |  |  |  | F I7 | D I7 | R I7 | E I7 | WB I7 |  | |  |  |  | F I8 | D I8 | R I8 | E I8 | WB I8 |  | |  |  |  |  | F I9 | D I9 | R I9 | E I9 | WB I9 | |  |  |  |  | F I10 | D I10 | R I10 | E I10 | WB I10 | |  |  |  |  |  |  |  |  | Timp  (*Zeit*) |   **Figura 6.** *Procesor superscalar (*Superskalarer Prozessor*)* | | |
| 2.2 Performanţele unui sistem de calcul |  | 2.2 Die Leistung eines Computersystems |
| 2.2.1 Viteza unui sistem de calcul |  | 2.2.1 Die Geschwindigkeit eines Computersystems |
| Pentru măsurarea vitezei calculatoarelor nu se poate folosi numai frecvenţa de ceas a CPU. Aceasta este o măsură neadecvată, întrucât este foarte importantă şi organizarea internă a procesoarelor.  S-a introdus apoi noţiunea de MIPS (*Millions of Instructions per Second*) – care nu mai depinde de frecvenţa tactului. Aceasta măsoară numărul (exprimat în milioane) de instrucţiuni (operând doar pe numere întregi) pe care le poate executa un procesor într-o secundă. MFLOPS (*Millions of Floating-Point Instructions per Second*) reprezintă numărul (exprimat în milioane) de instrucţiuni în virgulă flotantă pe care un procesor le poate executa în unitatea de timp.  În general, un microprocesor este caracterizat de:   * viteza de lucru; * capacitatea maximă de memorie pe care o poate adresa; * setul de instrucţiuni pe care le poate executa.   Ca şi criteriu de performanţă se consideră deseori viteza de lucru a CPU, care depinde de:   * frecvenţa ceasului intern, * capacitatea de paralelizare (organizarea execuţiei instrucţiunilor); * dimensiunea regiştrilor interni; * lăţimea de bandă a magistralei de date; * tipul microprocesorului sau dimensiunea memoriei cache a CPU (ca factor de influenţă a vitezei de comunicare cu exteriorul). |  | Zum Messen der Geschwindigkeit von Computern kann die CPU-Taktfrequenz nicht alleine verwendet werden. Dies ist eine unzureichende Maßnahme, da die interne Organisation der Prozessoren sehr wichtig ist.  Dann wurde der Begriff „Millionen von Anweisungen pro Sekunde“ (MIPS (*Millions of Instructions per Second*) eingeführt, der nicht mehr von der Taktfrequenz abhängt. Es misst die Anzahl (ausgedrückt in Millionen) von Anweisungen (die nur für Ganzzahlen ausgeführt werden), die ein Prozessor in einer Sekunde ausführen kann. MFLOPS (*Millions of Floating-Point Instructions per Second*) ist die Anzahl (in Millionen) von Gleitkomma-Anweisungen, die ein Prozessor in der Zeiteinheit ausführen kann.  Im Allgemeinen ist ein Mikroprozessor gekennzeichnet durch:  - Arbeitsgeschwindigkeit;  - die maximale Speicherkapazität, die es adressieren kann;  - den Befehlssatz, den es ausführen kann.  Als Leistungskriterium wird häufig die CPU-Geschwindigkeit herangezogen, die von Folgendem abhängt:   * die Frequenz der internen Uhr, * Fähigkeit zur Parallelisierung (Organisation der Ausführung von Anweisungen); * die Größe der internen Register; * die Bandbreite des Datenbusses; * den Typ des Mikroprozessors oder die Größe des CPU-Cache (als Faktor, der die Kommunikationsgeschwindigkeit nach außen beeinflusst). |
| 2.2.2 Criterii de evaluare a performanţelor |  | 2.2.2 Leistungsbewertungskriterien |
| În general sunt utilizate două tipuri de criterii de evaluare a performanţelor:   * timpul de execuţie (*Execution Time*) = timpul în care este executată o sarcină, timpul de răspuns; * rata de execuţie (*Throughput*) = numărul de sarcini rezolvate într-un anumit interval de timp.   Rezultă că îmbunătăţirea performanţei poate fi privită sub două aspecte:   * reducerea timpului de rezolvare a unei sarcini; * creşterea ratei de execuţie.   Acordarea unei „note” întregului sistem de calcul este dificil de realizat, motiv pentru care se face referire la diferite aspecte ale performanţelor componentelor sale. De exemplu:  - CPU:   * timpul de execuţie (perioadă a tactului mică, execuţie paralelă a instrucţiunilor); * rata de execuţie (MIPS, MFLOPS, planificarea eficientă a instrucţiunilor); * capacitatea memoriei cache a CPU * memoria cache: rata de transfer, hit rate vs. miss rate * memoria internă: capacitatea, rata de transfer * dispozitivele periferice: viteza de căutare, capacitatea de stocare, viteza de transfer, numărul de pixeli sau poligoane afişate pe secundă   - ş.a.m.d.  Alături de performanţele individuale ale componentelor, trebuie avute în vedere şi alte aspecte precum: compatibilitatea componentelor, tipurile de date gestionate sau de aplicaţii executate, sistemul de operare, disponibilitatea software-ului, costurile de proiectare sau costurile de achiziţionare sau întreţinere. |  | Im Allgemeinen werden zwei Arten von Leistungsbewertungskriterien verwendet:   * Ausführungszeit (*Execution Time*) = die Zeit, in der eine Aufgabe ausgeführt wird, die Antwortzeit; * Durchsatz (*Throughput*) = Anzahl der in einem bestimmten Zeitrahmen gelösten Aufgaben.   Es stellt sich heraus, dass die Leistungsverbesserung kannst auf zwei Arten angesehen sind:   * Verkürzung der Zeit zur Lösung einer Aufgabe; * Erhöhung der Ausführungsrate.   Das Gewähren einer „Note“ für das gesamte Computersystem ist schwierig zu erreichen, weshalb es sich auf verschiedene Aspekte der Leistung seiner Komponenten bezieht. Zu Beispiel:  - CPU:   * Ausführungszeit (niedrige Taktperiode, parallele Ausführung von Anweisungen); * Ausführungsrate (MIPS, MFLOPS, effektive Anweisungsplanung); * CPU-Cache-Kapazität * Cache-Speicher: Übertragungsrate, Trefferrate vs. Fehlrate * Interner Speicher: Kapazität, Übertragungsrate * Peripheriegeräte: Suchgeschwindigkeit, Speicherkapazität, Übertragungsgeschwindigkeit, Anzahl der pro Sekunde angezeigten Pixel oder Polygone   - und so weiter.  Neben den individuellen Leistungen der Komponenten sind weitere Aspekte wie die Kompatibilität der Komponenten, die Art der verwalteten Daten oder ausgeführten Anwendungen, das Betriebssystem, die Verfügbarkeit der Software, die Kosten für das Design oder die Kosten für den Kauf oder die Wartung zu berücksichtigen. |
| 2.3 „Dimensiunea” unui microprocesor |  | 2.3 Die „Größe“ eines Mikroprozessors |
| Există două perspective:   * perspectiva hardware: lăţimea de bandă a magistralei de date (de exemplu: Pentium are o magistrală de date pe 64 biţi = 64 linii de date, astfel că la fiecare *memory cycle* procesorul poate accesa 8 octeţi din memorie); * perspectiva software: dimensiunea unui cuvânt de memorie (dimensiunea regiştrilor CPU).   În multe cazuri cele două perspective au coincis ca dimensiune. Diferenţe de interpretare apar spre exemplu la procesoarele unde lăţimea de bandă a magistralei diferă de lăţimea registrelor interne. |  | Es gibt zwei Perspektiven:   * Hardware-Perspektive: die Bandbreite des Datenbusses (zum Beispiel: Pentium hat einen 64-Bit-Datenbus = 64 Datenleitungen, so dass der Prozessor bei jedem *memory cycle* – Speicherzyklus – auf 8 Byte Speicher zugreifen kann);   - Software-Perspektive: Die Größe eines Speicherworts (die Größe der CPU-Register).  In vielen Fällen stimmten die beiden Perspektiven überein. Interpretationsunterschiede treten beispielsweise bei Prozessoren auf, bei denen sich die Bandbreite des Busses von der Breite der internen Register unterscheidet. |
| 3. Memoria |  | 3. Der Speicher |
| Memoria este un dispozitiv de stocare a datelor pentru un anumit interval de timp.  Clasificări:  1. Din punct de vedere al accesării datelor:   * 1. *Memorie ROM* (Read-Only Memory), care poate fi accesată numai în citire;   2. *Memorie RAM, cu acces aleator* (*Random Access Memory*):locaţiile pot fi accesate atât în citire, cât şi în scriere;   3. *Memorie cu acces asociativ.* Exemplu: memoria cache.   Observaţie: La încărcarea sau căutarea în memoria cache a unei informaţii, pe baza adresei acesteia se determină care este poziţia din cache în care trebuie să fie încărcată sau la care ar trebui să se găsească dacă ar exista deja acolo. Strategia de determinare a locaţiei din memoria cache depinde de modul de organizare a acesteia. De exemplu: *maparea directă –* unei informaţii de la o anumită adresă îi corespunde o locaţie bine determinată din memoria cache, locaţie calculată cu ajutorul unei funcţii de transformare (poate fi funcţie de dispersie de tip modulo); *maparea asociativă pe mulţime –* o anumită informaţie poate fi încărcată în memoria cache în oricare din poziţiile dintr-o mulţime bine determinată.   * 1. *Memorie cu acces secvenţial*:pentru a accesa a *n*-a înregistrare, trebuie parcurse primele *n*-1 înregistrări => timpul de accesare a datelor este variabil, depinzând de locaţia accesată. Exemplu: benzi magnetice;   2. *Memorie cu acces direct*: spre deosebire de accesul secvenţial, poziţionarea pe o anumită înregistrare se face în mod direct pe baza unui calcul de adresă. Exemplu: dispozitivele de tip disc, precum hard disk, floppy disk, CD-ROM.   Observație: Modul de accesare a datelor din memoria cu acces aleator este similar cu cel folosit pentru memoria cu acces direct (direct, pe baza unei adrese cunoscute). Diferenţa constă în timpul de acces: în cazul memoriei cu acces aleator timpul de acces este acelaşi, indiferent de adresa datelor accesate; în schimb, memoria cu acces direct foloseşte un mecanism fizic de poziţionare la o anumită adresă, fapt care introduce un timp de întârziere în accesarea datelor => timp de procesare variabil, în funcţie de poziţia curentă a mecanismului de accesare şi de poziţia datelor pe disc.  2. Din punct de vedere al volatilităţii:  2.1 *Memorie volatilă* (de scurtă durată): conţinutul său se pierde la îndepărtarea sursei de curent. Aici se încadrează memoria principală a SC (care conţine datele şi instrucţiunile utilizate curent de CPU);  2.2 *Memorie non-volatilă sau remanentă* (de lungă durată): conţinutul se păstrează şi după deconectarea de la sursă. Exemple: memoria ROM, hard disk, CDROM, memoria Flash.  3. Din punct de vedere al accesului CPU:  3.l. *Memorie internă*: accesată direct de către CPU;  3.2. *Memorie secundară* sau *dispozitiv de stocare periferic*: memorie externă, cu acces indirect la CPU. Exemple: HD, floppy disk, CDROM.  4. Din punct de vedere al tipurilor de acces permise:  4.1 *Memorie read / write*: permite acces la date în citire sau scriere. Exemple: memoria principală, hard disk, floppy disk;  4.2. *Memorie read-only*: permite doar citirea datelor. Exemple: ROM, CDROM. |  | Der Speicher ist ein Gerät, der die Daten für eine bestimmte Zeit speichert.  Klassifizierung:  1. Aus Sicht des Datenzugriffs:   * 1. *Nur-Lese-Speicher* (*Read-Only Memory* ROM), der nur gelesen werden kann;   2. *Direktzugriffsspeicher* (*Random Access Memory* RAM): Auf Speicherorte kann sowohl beim Lesen als auch beim Schreiben zugegriffen werden.   3. *Speicher mit assoziativem Zugriff*. Beispiel: Cache-Speicher.   Hinweis: Beim Hochladen oder Zwischenspeichern von Informationen wird basierend auf ihrer Adresse festgelegt, an welche Cache-Position sie hochgeladen werden sollen oder wo sie sich befinden sollen, wenn sie bereits vorhanden sind. Die Strategie zur Bestimmung des Cache-Speicherorts hängt davon ab, wie sie organisiert ist. Zum Beispiel: *Direkte Zuordnung* – Informationen von einer bestimmten Adresse entsprechen einem gut bestimmten Cache-Speicherort, einem Speicherort, der unter Verwendung einer Transformationsfunktion berechnet wurde (kann eine Funktion der Dispersion vom Modulo-Typ sein); *Assoziatives Mapping auf dem Set* – Eine bestimmte Information kann an jeder Position in einem Set in den Cache Speicher geladen werden.   * 1. *Speicher mit sequenziellem Zugriff*: Um auf den *n*-ten Datensatz zuzugreifen, müssen die ersten *n*-1 Aufzeichnungen abgeschlossen sein => die Datenzugriffszeit ist variabel, abhängig vom Speicherort, auf den zugegriffen wird. Beispiel: Magnetstreifen;   2. *Direktzugriffsspeicher*: Im Gegensatz zum sequentiellen Zugriff erfolgt die Positionierung auf einem bestimmten Datensatz direkt anhand einer Adressberechnung. Beispiel: Datenträgertypen wie Festplatte, Diskette, CD-ROM.   Hinweis: Der Datenzugriff aus dem Direktzugriffsspeicher erfolgt auf ähnliche Weise wie der Direktzugriffsspeicher (direkt, basierend auf einer bekannten Adresse). Der Unterschied besteht in der Zugriffszeit: Im Fall eines Direktzugriffsspeichers ist die Zugriffszeit unabhängig von der Adresse der zugegriffenen Daten gleich. Im Gegensatz dazu verwendet der Direktzugriffsspeicher einen physikalischen Positionierungsmechanismus an einer bestimmten Adresse, der eine Zeitverzögerung beim Zugriff auf Daten einführt => variable Verarbeitungszeit, abhängig von der aktuellen Position des Zugriffsmechanismus und der Position der Daten auf der Platte.  2. In Bezug auf die Volatilität:  2.1 *Flüchtiges (Kurzzeit-) Speicher*: Der Inhalt geht verloren, wenn die Stromquelle entfernt wird. Hier befindet sich der Hauptspeicher des CS (der die aktuell von der CPU verwendeten Daten und Anweisungen enthält);  2.2 *Nichtflüchtiger oder verbleibender (Langzeit-) Speicher*: Der Inhalt bleibt nach dem Trennen von der Quelle erhalten. Beispiele: ROM, Festplatte, CD-ROM, Flash-Speicher.  3. Aus Sicht des CPU-Zugriffs:  3.l. *Interner Speicher*: Direktzugriff durch die CPU;  3.2. *Sekundärspeicher oder Peripheriespeicher*: Externer Speicher mit indirektem CPU-Zugriff. Beispiel: HD, Diskette, CD-ROM.  4. Unter dem Gesichtspunkt der zulässigen Zugriffsarten:  4.1 *Lese- / Schreibspeicher*: Ermöglicht den Zugriff auf Daten beim Lesen oder Schreiben. Beispiele: Hauptspeicher, Festplatte, Diskette;  4.2. *Nur-Lese-Speicher*: Ermöglicht nur das Lesen von Daten. Beispiel: ROM, CD-ROM. |

|  |  |  |
| --- | --- | --- |
| 3.1 Memoria internă |  | 3.1 Interner Speicher |
| Memoria internă reprezintă toate spaţiile de stocare de date accesibile CPU fără utilizarea canalelor de comunicaţie de intrare / ieşire. Din această categorie fac parte memoria principală, memoria *cache* (dintre CPU şi memoria principală), ROM şi registrele, toate aceste dispozitive putând fi direct accesate de către CPU.  Memoria principală (*main memory, primary memory*), cunoscută în general sub numele de memorie RAM, conţine date care sunt utilizate în mod curent de către procesor – *instrucţiuni* ale programelor care sunt executate şi *date* cu care acestea operează. Aceste informaţii sunt aduse în memoria principală de pe un suport de stocare extern sau de la un dispozitiv de I/O. Tipurile de memorie care sunt folosite ca memorie principală sunt cele din clasa memoriilor RAM, precum DRAM sau SRAM.  *Memoria DRAM* (*Dynamic RAM*) face parte din clasa memoriilor volatile. Datorită modului în care este construită, este necesară reactualizarea conţinutului la un anumit interval de timp, de exemplu de ordin μs (de aici denumirea „dinamică”). Datele nu sunt disponibile în timpul operaţiilor de reactualizare. Deşi timpul consumat de aceste operaţii constituie aproximativ 1% din timpul de funcţionare, viteza de acces este mai redusă faţă de alte tipuri de memorie. Conţinutul unei asemenea memorii este organizat ca tablou bidimensional de biţi. La citirea unui element al tabloului, se citeşte întreg rândul, care apoi este rescris (*refresh*). Pentru operaţia de scriere a unui element, se citeşte întreg rândul, se modifică elementul, apoi se rescrie întreg rândul înapoi. Elementele unei memorii DRAM sunt mai mici şi mai ieftine decât elementele SRAM.  Observaţie: O alternativă a memoriei DRAM ca organizare este memoria flash, întâlnită în prezent în dispozitive precum carduri de memorie, dispozitive flash USB, camere digitale, telefoane mobile. Acest tip de memorie are un cost per bit mai mic decât al memoriei DRAM, este non-volatilă, dar de viteză mai mică la citire / scriere.  *Memoria SRAM* (*Static RAM*) este un tip de memorie volatilă. După cum indică denumirea, conţinutul unei memorii SRAM se păstrează atâta timp cât sistemul este conectat la o sursă, spre deosebire de DRAM care necesită reactualizări periodice ale conţinutului. Structura SRAM permite un acces mai rapid la locaţiile acesteia, în comparaţie cu DRAM, motiv pentru care este utilizată ca memorie cache a CPU. Memoriile SRAM de viteză şi capacitate mai mică sunt folosite atunci când se cere un consum de energie şi cost scăzut, de exemplu pentru backup RAM cu sursă de tip baterie. Deoarece este mai puţin densă faţă de DRAM (conţine mai puţini biţi pe unitate de suprafaţă), în general capacitatea unei memorii SRAM este mai mică faţă de unei memorii DRAM.  Observaţie: De obicei, raportul de capacitate DRAM/SRAM = 4-8; raportul de cost şi timp de acces SRAM/DRAM = 8-16.  Deoarece nu poate fi (uşor) scrisă, memoria ROM este utilizată în general ca spaţiu de stocare al *firmware*-ului, care nu necesită actualizări frecvente. *Firmware* reprezintă un ansamblu de resurse software care, prin intermediul microprogramării, preia unele funcţii care în mod tradiţional revin software-ului (sistemului de operare),  Memoria ROM a multor sisteme de calcul din generaţiile trecute (anii '80) conţinea încă de la furnizare sistemul de operare, iar o parte din acestea includeau şi un interpretor al limbajului de programare BASIC. În prezent, tendinţa este de a stoca cât mai puţine informaţii în memoriile ROM şi o cantitate tot mai mare de date pe dispozitivele de memorare externe. Deşi memoria ROM este de capacitate mică, avantajul principal este viteza mare de accesare a datelor. În general este întâlnită ca şi componentă CPU, caz în care conţine programul de control al acestuia, sau ca suport pentru BIOS. BIOS-ul (*Basic Input/Output System*) este un set de rutine de nivel scăzut care sunt responsabile de iniţializarea sistemului, verificarea echipamentelor periferice din sistem şi accesul primar la acestea. BIOS-ul poate fi considerat şi *firmware*-ul plăcii de bază, rutinele sale fiind printre primele care se execută la pornirea unui calculator.  Memoria ROM este în general cunoscută ca memorie scrisă în faza de producţie, al cărui conţinut nu poate fi modificat ulterior. Mai există, însă, câteva alte tipuri de memorie ROM al căror conţinut poate fi rescris, precum:   * PROM (*Programmable Read-Only Memory*) – poate fi scrisă (programată) o singură dată cu ajutorul unui echipament specializat; * EPROM (*Erasable Programmable Read-Only Memory*) – permite ştergerea conţinutului (prin expunere la ultraviolete) şi rescrierea acestuia cu ajutorul unui programator; * EAROM (*Electrically Alterable Read-Only Memory*) – este folosită în general pentru memorarea permanentă a unor parametri ai sistemului, motiv pentru care este rar modificată; la un moment dat, poate fi alterată o parte a conţinutului; * EEPROM (*Electrically Erasable Read-Only Memory*) – permite ştergerea electrică a întregului conţinut sau doar a unui bloc din conţinutul memoriei şi rescrierea acestuia; exemplu: memoria flash a camerelor digitale, MP3 player-elor ş.a. |  | Der interner Speicher ist der gesamte für die CPU zugängliche Datenspeicherplatz ohne Verwendung von Ein- / Ausgabekommunikationskanälen. Diese Kategorie umfasst Hauptspeicher, *Cachespeicher* (zwischen CPU und Hauptspeicher), ROMs und Register, auf die die CPU direkt zugreifen kann.  Der Hauptspeicher (*main memory, primary memory*), allgemein als RAM bezeichnet, enthält Daten, die üblicherweise vom Prozessor verwendet werden - *Anweisungen* für die ausgeführten Programme und *Daten*, mit denen sie arbeiten. Diese Informationen werden von einem externen Speichermedium oder einem E / A-Gerät in den Hauptspeicher übertragen. Die Arten von Speicher, die als Hauptspeicher verwendet werden, sind diejenigen in der RAM-Klasse, wie z. B. DRAM oder SRAM.  *DRAM-Speicher* (*Dynamic RAM*) gehört zur Klasse der flüchtigen Speicher. Aufgrund der Art der Erstellung ist es erforderlich, den Inhalt zu einem bestimmten Zeitpunkt zu aktualisieren, beispielsweise zu μs (daher der Begriff „dynamisch“). Während der Upgrade-Vorgänge sind die Daten nicht verfügbar. Obwohl die von diesen Vorgängen verbrauchte Zeit etwa 1% der Betriebszeit ausmacht, ist die Zugriffsgeschwindigkeit langsamer als bei anderen Arten von Speichern. Der Inhalt eines solchen Speichers ist als zweidimensionales Array von Bits organisiert. Beim Lesen eines Elements des Gemäldes wird die gesamte Zeile gelesen, die dann neu geschrieben wird (*refresh*). Um ein Element zu schreiben, lesen Sie die gesamte Zeile, ändern Sie das Element und schreiben Sie die gesamte Zeile zurück. DRAM-Speicherelemente sind kleiner und billiger als SRAM-Elemente.  Hinweis: Eine Alternative zum DRAM-Speicher als Organisation ist der Flash-Speicher, der derzeit in Geräten wie Speicherkarten, USB-Flash-Geräten, Digitalkameras und Mobiltelefonen verwendet wird. Diese Art von Speicher hat geringere Kosten pro Bit als DRAM-Speicher, ist nichtflüchtig, jedoch langsamer bei Lese- / Schreibgeschwindigkeit.  *SRAM-Speicher* (*Static RAM*) ist eine Art flüchtiger Speicher. Wie der Name andeutet, wird der Inhalt eines SRAM-Speichers so lange gespeichert, wie das System mit einer Quelle verbunden ist, im Gegensatz zu DRAM, das regelmäßige Aktualisierungen des Inhalts erfordert. Die SRAM-Struktur ermöglicht im Vergleich zu DRAM einen schnelleren Zugriff auf ihre Speicherorte, weshalb sie als CPU-Cache verwendet wird. SRAM-Speicher mit niedriger Geschwindigkeit und niedriger Kapazität werden verwendet, wenn ein geringer Stromverbrauch und Kosten erforderlich ist, z. B. für die RAM-Sicherung mit Batteriequelle. Da es weniger dicht ist als DRAM (enthält weniger Bits pro Oberflächeneinheit), ist die Kapazität eines SRAM-Speichers im Allgemeinen kleiner als die eines DRAM-Speichers.  Hinweis: Typischerweise ist das DRAM / SRAM-Kapazitätsverhältnis = 4-8; Kosten- und Zugriffszeitverhältnis SRAM / DRAM = 8-16.  Da es nicht (einfach) beschrieben werden kann, wird das ROM im Allgemeinen als *Firmwares* Speicher verwendet, für den keine häufigen Aktualisierungen erforderlich sind. *Firmware* ist ein Satz von Softwareressourcen, die durch Mikroprogrammierung einige Funktionen übernehmen, die traditionell zur Software (Betriebssystem) gehören.  Der ROM-Speicher vieler Rechnersysteme früherer Generationen (1980er Jahre) enthielt bereits das Betriebssystem aus dem Lieferumfang und zum Teil auch einen Interpreter der Programmiersprache BASIC. Gegenwärtig besteht die Tendenz, so wenige Informationen wie möglich in ROMs und eine zunehmende Datenmenge auf externen Speichergeräten zu speichern. Obwohl der ROM-Speicher eine geringe Kapazität aufweist, ist der Hauptvorteil eine hohe Datenzugriffsgeschwindigkeit. Es wird im Allgemeinen als CPU-Komponente gefunden, in welchem Fall es sein Steuerprogramm enthält, oder als BIOS-Unterstützung. Das BIOS (*Basic Input / Output System*) besteht aus einer Reihe von Routinen auf niedriger Ebene, die für die Systeminitialisierung, die Überprüfung der Peripheriegeräte im System und den primären Zugriff darauf verantwortlich sind. Das BIOS kann auch als *Firmware* des Hauptplatines betrachtet werden, wobei seine Routinen zu den ersten gehören, die beim Starten eines Computers ausgeführt werden.  Der ROM-Speicher wird in der Produktionsphase allgemein als geschriebener Speicher bezeichnet, dessen Inhalt später nicht mehr geändert werden kann. Es gibt jedoch einige andere Arten von ROM-Speichern, deren Inhalt umgeschrieben werden kann, beispielsweise:   * PROM (*Programmable Read Only Memory*) – kann nur einmal mit Hilfe von Spezialgeräten beschrieben (programmiert) werden; * EPROM (*Erasable Programmable Read Only Memory*) – Ermöglicht das Löschen des Inhalts (durch Bestrahlung mit Ultraviolett) und dessen Neuschreiben mit Hilfe eines Programmiergeräts. * EAROM (*Electrically Alterable Read Only Memory*) – wird im Allgemeinen zur dauerhaften Speicherung von Systemparametern verwendet, weshalb es selten geändert wird. Irgendwann kann ein Teil des Inhalts geändert werden. * EEPROM (*Electrically Erasable Read-Only Memory*) – Ermöglicht das elektrische Löschen des gesamten Inhalts oder nur eines Inhaltsblocks des Speichers und dessen Neuschreiben. Beispiel: Flash-Speicher von Digitalkameras, MP3-Playern usw. |
| 3.2 Memoria externă / secundară |  | 3.2 Externer / sekundärer Speicher |
| Memoria secundară reprezintă un dispozitiv de stocare pe termen lung a datelor, care nu sunt curent folosite de către CPU. În general este de capacitate mai mare şi are o viteză mai mică de accesare a datelor faţă de memoria internă şi face parte din categoria memoriilor non-volatile.  Câteva dispozitive incluse în această categorie de memorie sunt: hard disk (HDD), floppy disk (FDD), compact disc (CD), DVD, banda magnetică, memoria flash.  *Structura unui volum disc*  Din punct de vedere fizic, un dispozitiv (volum) disc poate fi alcătuit din unul sau mai multe discuri, plasate concentric pe un ax, în jurul căruia se rotesc cu o viteză constantă. Informaţia poate fi înregistrată magnetic pe una sau ambele feţe ale unui disc. Pentru accesarea informaţiei, fiecare suprafaţă de memorare are asociat un cap de citire / scriere. Braţele capetelor de accesare corespunzătoare discurilor sunt situate pe un suport unic al dispozitivului. Mişcarea acestora permite deplasarea capetelor de acces radial pe suprafaţa discului, permiţând astfel accesarea informaţiei indiferent de localizarea acesteia faţă de axul central.  Din punct de vedere logic, o suprafaţă de memorare a discului este divizată în benzi concentrice numite *piste*. Pentru un dispozitiv care deţine mai multe suprafeţe de memorare, numărul de piste de pe aceste suprafeţe este acelaşi, iar pistele de aceeaşi rază formează un cilindru. O pistă este divizată în porţiuni numite sectoare. Numărul de sectoare este acelaşi pentru fiecare pistă a discului şi fiecare sector are aceeaşi dimensiune. Un sector reprezintă în general unitatea de transfer de date între disc şi memoria internă.  Parametrii dispozitivului disc sunt următorii: numărul de discuri, numărul de capete de citire / scriere pentru un disc (numărul de feţe active ale unui disc), numărul de piste de pe o faţă, numărul de sectoare de pe o pistă şi numărul de octeţi de pe un sector.  O metodă de adresare a informaţiilor de pe disc utilizată este CHS (*cylinder-head-sector*). Identificarea sectorului accesat se realizează prin specificarea numărului de ordine al capului de citire / scriere, numărul cilindrului şi numărul sectorului din cadrul pistei (unde pista e determinată de cap şi cilindru).  Un criteriu de evaluare a performanţei unui dispozitiv disc este timpul de accesare a informaţiei, care poate fi calculat după formula:  *TimpAcces = TimpCăutare + TimpRotire + TimpTransfer*,  unde:   * TimpCăutare depinde de numărul de piste peste care trebuie să de deplaseze capul de accesare şi viteza de căutare a discului. Aici putem adăuga şi TimpControler, care reprezintă întârzierea produsă de controlerul dispozitivului (acesta având rol de interfaţă de comunicare între CPU şi disc); * TimpRotire este în funcţie de viteza de rotaţie a discului şi de distanţa dintre sectorul care trebuie accesat şi capul de accesare; * TimpTransfer depinde de rata de transfer a datelor (*bandwidth*) caracteristică dispozitivului. |  | Der Sekundärspeicher ist ein Langzeitdatenspeicher, der von der CPU nicht häufig verwendet wird. Es hat im Allgemeinen eine größere Kapazität und eine geringere Datenzugriffsgeschwindigkeit als der interne Speicher und gehört zur Kategorie der nichtflüchtigen Speicher.  Einige Geräte in dieser Kategorie von Speicher sind: Festplatte (HDD), Diskette (FDD), CD (CD), DVD, Magnetband, Flash-Speicher.  *Die Struktur eines Datenträgers*  Aus physikalischer Sicht kann eine Geräte- (Volumen-) Platte aus einer oder mehreren Scheiben bestehen, die konzentrisch auf einer Achse angeordnet sind und um die sie sich mit konstanter Geschwindigkeit drehen. Die Informationen können auf einer oder beiden Seiten einer Disc magnetisch aufgezeichnet werden. Um auf die Informationen zuzugreifen, ist jeder Speicheroberfläche ein Lese- / Schreibkopf zugeordnet. Die Zubehörarme der Discs befinden sich auf einer einzigen Halterung des Geräts. Durch ihre Bewegung können sich die Enden des radialen Zugriffs auf der Oberfläche der Platte bewegen, wodurch der Zugriff auf Informationen unabhängig von ihrer Position relativ zur Mittelachse ermöglicht wird.  Logischerweise ist eine Plattenspeicherfläche in konzentrische Bänder unterteilt, die als *Spuren* bezeichnet werden. Bei einem Gerät mit mehreren Speicherflächen ist die Anzahl der Spuren auf diesen Flächen gleich und die Spuren mit dem gleichen Radius bilden einen Zylinder. Eine Spur ist in Abschnitte unterteilt, die als Sektoren bezeichnet werden. Die Anzahl der Sektoren ist für jede Plattenspur gleich und jeder Sektor hat die gleiche Größe. Ein Sektor ist im Allgemeinen die Einheit für die Datenübertragung zwischen Festplatte und internem Speicher.  Die Plattengeräteparameter lauten wie folgt: Anzahl der Platten, Anzahl der Lese- / Schreibenden für eine Platte (Anzahl der aktiven Seiten einer Platte), Anzahl der Spuren auf einer Seite, Anzahl der Sektoren auf einer Spur und Anzahl der Bytes von auf einem Sektor.  Ein Verfahren zum Adressieren der verwendeten Platteninformation ist CHS (*cylinder-head-sector*, Zylinderkopfsektor). Die Identifizierung des Sektors, auf den zugegriffen wird, erfolgt durch Angabe der Ordnungsnummer des Lese- / Schreibkopfs, der Nummer des Zylinders und der Nummer des Sektors innerhalb der Spur (wobei die Spur durch den Kopf und den Zylinder bestimmt wird).  Ein Kriterium für die Bewertung der Leistung eines Plattengeräts ist der Zeit des Zugriffs auf die Informationen, der sich nach folgender Formel berechnen lässt:  *Zugriffszeit* = *Suchzeit* + *Rotationszeit* + *Übertragungszeit*,  wo:   * Die Suchzeit hängt von der Anzahl der Titel ab, über die sich der Zugriffskopf und die Suchgeschwindigkeit der Festplatte bewegen müssen. Hier können wir auch ControllerZeit hinzufügen, der die Verzögerung darstellt, die durch den Gerätecontroller verursacht wird (dieser hat die Rolle der Kommunikationsschnittstelle zwischen der CPU und der Festplatte). * Rotationszeit basiert auf der Rotationsgeschwindigkeit der Platte und der Entfernung zwischen dem Sektor, auf den zugegriffen werden soll, und dem Zugriffskopf. * Übertragungszeit ist abhängig von der Datenübertragungsrate (*bandwidth*, Bandbreite) des Geräts. |
| enter image description herehttps://i.stack.imgur.com/2plwL.png  **Figura 7.** *Structura fizică a unui hard disc (*Physikalische Struktur einer Festplatte*)* | | |
| Cele mai utilizate volume disc sunt:   * hard disk-urile – dispozitive de stocare a datelor pe suport magnetic, alcătuite din mai multe discuri; capacitatea de memorare a acestora este de ordinul sutelor de gigabytes; * benzile magnetice – permit acces secvenţial la date, asigură o capacitate mare de stocare, sunt ieftine şi sunt folosite în general pentru arhivare şi memorarea copiilor de siguranţă (suport de backup);   - discurile optice:   * CD (Compact Disc) – dispozitive de capacitate de memorare relativ mare (650-700 MB), de tip Read-Only sau Read / Write, pentru care producţia şi duplicarea nu sunt costisitoare; * DVD (*Digital Versatile Disc*) – dispozitive disc similare CD-urilor, însă prezintă un mod de codificare a datelor diferit şi o densitate a acestora mai mare; capacitatea de stocare a acestora este în prezent de 4.7 GB până la 17.1 GB; pot fi de tip Read Only (DVD-ROM) sau Read / Write (DVD-RW); * Alte tipuri de discuri optice: Blu-Ray Disc, High Density Digital Versatile Disc (HD DVD) etc. |  | Die am häufigsten verwendeten Datenträger sind:   * Festplatten - Geräte zum Speichern von Daten auf magnetischen Medien, die aus mehreren Festplatten bestehen; Ihre Speicherkapazität liegt in der Größenordnung von Hunderten von Gigabyte. * Magnetstreifen – Ermöglichen den sequentiellen Zugriff auf Daten, bieten eine hohe Speicherkapazität, sind kostengünstig und werden in der Regel zum Archivieren und Speichern von Backups (Backup-Medien) verwendet. * optische Disks: * CD (Compact Disc) – Geräte mit relativ großer Speicherkapazität (650-700 MB) vom Typ Read-Only oder Read / Write, für die Produktion und Vervielfältigung nicht teuer sind; * DVD (*Digital Versatile Disc*) – Geräte, die CDs ähneln, jedoch einen anderen Datencodierungsmodus und eine höhere Dichte aufweisen. Ihre Speicherkapazität beträgt derzeit 4,7 GB bis 17,1 GB. Sie können Read Only (DVD-ROM) oder Read / Write (DVD-RW) sein. * Andere Arten von optischen Discs: Blu-Ray-Discs, High Density Digital Versatile Discs (HD-DVDs) usw. |
| 3.3 Ierarhia memoriei |  | 3.3 Die Speicherhierarchie |
| Diferenţa de performanţă dintre componentele unui SC poate fi semnificativă. Pentru a gestiona cât mai eficient accesul la date, un sistem de calcul deţine un sistem complex de memorie, în care combină memorii de capacitate mică, dar rapide, şi memorii de capacitate mare, însă de viteză redusă. Drept rezultat, un asemenea sistem se comportă în general ca o memorie rapidă, de capacitate mare. Nivelurile ierarhice ale unui asemenea sistem pot fi reprezentate astfel (Figura 8): |  | Der Leistungsunterschied zwischen den Komponenten eines CS kann erheblich sein. Um den Datenzugriff so effizient wie möglich zu verwalten, verfügt ein Computersystem über ein komplexes Speichersystem, in dem kleiner, aber schneller Speicher und großer, aber langsamer Speicher kombiniert werden. Infolgedessen verhält sich ein solches System im Allgemeinen wie ein schneller Speicher mit hoher Kapazität. Die hierarchischen Ebenen eines solchen Systems können wie folgt darstellen (Abbildung 8): |
| |  | | --- | | Register | | „On-chip” (L1) Cachespeicher | | „On-chip” (L2) Cachespeicher | | „On-board” (L3) Cachespeicher | | Hauptspeicher | | Sekundärer Speicher | | Tertiäres Speicher |   Erhöht den Wert der Kosten / Speichereinheit  Erhöht die Speicherkapazität  Erhöhen die Zugriffsgeschwindigkeit  **Figura 8.** *Ierarhia memoriei (*Die Speicherhierarchie*)* | | |
| Se observă că nivelurile de memorie de capacitate mai mică se găsesc mai aproape de procesor decât memoriile de capacitate mare, dar de viteză de acces mai mică.  În general, un nivel al ierarhiei reprezintă o submulţime de informaţii a unui nivel inferior: datele care se găsesc pe primul nivel sunt aduse din următorul nivel de memorie, mai îndepărtat de CPU. Pentru a gestiona un asemenea trafic al datelor între diferite niveluri este nevoie de funcţii de transformare a adreselor de pe nivelul inferior către cel imediat superior. Întotdeauna datele sunt copiate numai între două niveluri adiacente.  Eficienţa unui asemenea sistem este asigurată de *principiul localizării*:   * *localizare temporală*: după accesarea unei date sunt mari şanse ca ea să fie accesată din nou în scurt timp => ar trebui să se mai reţină data respectivă pentru o perioadă de timp (de exemplu: instrucţiunile dintr-o structură repetitivă sau ale unei subrutine); * *localizare spaţială*: dacă se accesează o locaţie, sunt mari şanse să urmeze accesarea unor locaţii din vecinătatea primeia => ar trebui ca la accesarea datei curente să se aducă un întreg bloc de informaţie care să conţină atât informaţia necesară în momentul curent, cât şi informaţia conţinută la adrese învecinate (de exemplu: variabile locale unei subrutine sau elementele unui şir).   În urma acestor observaţii statistice s-au dezvoltat aşa-numitele memorii de tip cache. |  | Es ist zu erkennen, dass Speicherebenen mit geringerer Kapazität näher am Prozessor liegen als Speicher mit hoher Kapazität, jedoch mit geringerer Zugriffsgeschwindigkeit.  Im Allgemeinen repräsentiert eine Hierarchieebene eine Teilmenge von Informationen einer niedrigeren Ebene: Die auf der ersten Ebene gefundenen Daten werden von der nächsten Speicherebene weiter von der CPU entfernt gebracht. Um diesen Datenverkehr zwischen verschiedenen Ebenen zu verwalten, ist es erforderlich, die Adressen von der niedrigeren auf die unmittelbar höhere Ebene umzuwandeln. Die Daten werden immer nur zwischen zwei benachbarten Ebenen kopiert.  Die Effizienz eines solchen Systems wird durch *das Prinzip der Lokalisierung* sichergestellt:   * *Temporärer Speicherort*: Nach dem Zugriff auf Daten besteht die Möglichkeit, dass in kurzer Zeit erneut auf die Daten zugegriffen wird. => Das Datum sollte für einen bestimmten Zeitraum beibehalten werden (z. B. Anweisungen in einer sich wiederholenden Struktur oder einem Unterprogramm); * *räumlicher Lage*: Wenn auf einen Ort zugegriffen wird, ist es wahrscheinlich, dass die Zugriffe auf Orte in der Nähe des ersten folgen => es sollte auf das aktuelle Datum gebracht werden, um einen vollständigen Informationsblock zu bringen, der sowohl die zum aktuellen Zeitpunkt als auch die benötigten Informationen enthält Informationen, die in benachbarten Adressen enthalten sind (z. B. lokale Variablen eines Unterprogramms oder Elemente eines Strings).   Infolge dieser statistischen Beobachtungen entwickelten sich die sogenannten Cache-Speicher. |
| 3.3.1. Memoria cache |  | 3.3.1 Der Cache-Speicher |
| O memorie de tip cache este o colecţie de date care reprezintă duplicarea valorilor originale stocate într-un alt tip de dispozitiv de memorare, a căror accesare pentru citire / procesare este mai costisitoare (ca timp) decât accesarea lor din cache. Memoria cache are capacitate mai mică, însă oferă un timp de acces la date cu mult mai rapid faţă de timpul asigurat de componenta asociată.  Odată ce datele sunt aduse în memoria cache, ele vor fi accesate de aici, fără a fi nevoie de repetarea operaţiei de copiere, scăzând astfel semnificativ timpul mediu de accesare.  Memoria cache exploatează localitatea spaţială şi temporală. O putem întâlni ca interfaţă între diferite niveluri ale ierarhiei memoriei sau în asociere cu diferite dispozitive periferice sau chiar componente soft. Astfel, în prezent noţiunea de memorie cache reprezintă o tehnică de optimizare a accesului la date, indiferent de tipul clientului cache pe care îl deserveşte (memorie, dispozitiv periferic sau componentă software – sistem de operare sau aplicaţie utilizator). În general, însă, se face referinţă la memoria cache ca fiind interfaţa dintre CPU şi memoria principală.  Exemplu: interfaţa dintre CPU şi memoria internă de tip DRAM. Aceasta poate fi alcătuită din unul, două sau trei niveluri de memorie cache. O mare parte a sistemelor de calcul din prezent folosesc memorie cache pe două niveluri (L1 şi L2). Primul nivel de cache este integrat pe chip-ul CPU (cache „on-chip”) şi asigură o viteză de acces similară CPU. Al doilea nivel asigură interfaţa dintre primul nivel şi memoria internă şi în general este memorie de tip SRAM. Este plasată de obicei tot pe chip-ul CPU, însă viteza de acces este mai redusă decât cea asigurată L1 cache, iar capacitatea de stocare este mai mare. În cazul în care există şi un al treilea nivel de memorie cache (L3), aceasta este amplasată pe placa de bază (cache „on-board”).  Secvenţa de căutare este:   * CPU solicită o instrucţiune sau un operand; să notăm cu *I* informaţia solicitată. * Se caută în primul nivel cache, cel mai apropiat de CPU. * Dacă *I* este găsită => *cache hit* (succes); se opreşte căutarea pe acest nivel. * Dacă *I* nu este găsită => *cache miss* (eşec); se continuă căutarea pe nivelul cache secundar * În cazul în care pe toate nivelurile cache s-a raportat eşec, se caută I în memoria principală.   Criterii de măsură a performanţei unei memorii cache: *hit rate* = nr\_hit / nr\_referinţe la memorie (procentul de accesări cu succes din totalul de accesări); *miss rate* = nr\_miss / nr\_referinţe la memorie (procentul de tentative de acces eşuate din totalul de cereri), unde *miss rate* = 1 - *hit rate*.  Principalele tipuri de memorie cache:   * cache al memoriei principale, ca interfaţă între aceasta şi CPU: poate fi pe unul, două sau trei niveluri; acest cache este gestionat de către hardware; * cache între memoria principală şi memoria secundară (disc) este memoria virtuală; transferul datelor de pe disc în memoria principală (gestiunea memoriei virtuale) este gestionat de sistemul de operare; * cache *Translation Lookaside Buffer* (TLB) al tabelei de pagini folosită pentru realizarea corespondenţei de adrese între memoria principală şi memoria virtuală; * memorii cache gestionate de componente soft: cache DNS (pentru corespondenţe dintre nume de domenii şi adrese IP); cache al unui web browser (pentru ultimele pagini accesate); cache al unui SGBD (de exemplu: Oracle, SQL-Server, pentru ultimele date citite sau ultimele planuri de execuţie dezvoltate).   În general, performanţa unei memorii cache depinde de parametrii de organizare şi funcţionare ai acesteia, precum: algoritmul de plasare a blocurilor aduse în cache (*block placement strategy*), modalitatea de identificare a unui bloc din cache (*block identification policy*), politica de selectare a unui bloc care să fie înlocuit în cazul în care nu mai este spaţiu liber în cache (*block replacement policy*), scrierea datelor modificate (*cache write policy* – imediat sau la dealocarea blocului). |  | Ein Cache-Speicher ist eine Sammlung von Daten, die ein Duplikat der Originalwerte darstellt, die in einem anderen Typ von Speichergerät gespeichert sind, dessen Lese- / Verarbeitungszugriff zeitaufwendiger ist als der Cache-Speicher Zugriff. Der Cache-Speicher hat eine geringere Kapazität, bietet jedoch eine schnellere Zugriffszeit auf Daten als die von der zugeordneten Komponente bereitgestellte Zeit.  Sobald die Daten in dem Cache-Speicher sind, wird von hier aus auf sie zugegriffen, ohne dass der Kopiervorgang wiederholt werden muss, wodurch die durchschnittliche Zugriffszeit erheblich verkürzt wird.  Der Cache-Speicher nutzt die räumliche und zeitliche Lokalität aus. Wir können es als Schnittstelle zwischen verschiedenen Ebenen der Speicherhierarchie oder in Kombination mit verschiedenen Peripheriegeräten oder sogar Softwarekomponenten finden. Daher ist der Cache-Speicher Begriff gegenwärtig eine Technik zur Optimierung des Zugriffs auf Daten, unabhängig vom Typ des Cache-Clients, den er bedient (Speicher, Peripheriegerät oder Softwarekomponente - Betriebssystem oder Benutzeranwendung). Im Allgemeinen wird der Cache-Speicher jedoch als Schnittstelle zwischen der CPU und dem Hauptspeicher bezeichnet.  Beispiel: Die Schnittstelle zwischen der CPU und dem internen DRAM-Speicher. Dies kann aus einer, zwei oder drei Ebenen des Cache-Speichers bestehen. Die meisten heutigen Computersysteme verwenden einen zweistufigen Cache (L1 und L2). Die erste Cache-Ebene ist auf dem CPU-Chip integriert („On-Chip“ -Cache) und bietet eine ähnliche CPU-Zugriffsgeschwindigkeit. Die zweite Ebene bildet die Schnittstelle zwischen der ersten Ebene und dem internen Speicher und ist im Allgemeinen ein Speicher vom Typ SRAM. Es wird normalerweise auch auf dem CPU-Chip platziert, aber die Zugriffsgeschwindigkeit ist langsamer als der gesicherte L1-Cache und die Speicherkapazität ist höher. Wenn es eine dritte Cache-Ebene (L3) gibt, befindet sie sich auf der Hauptplatine („On-Board“ -Cache).  Die Suchreihenfolge ist:   * Die CPU fordert eine Anweisung oder einen Operanden an. Wir notieren mit *I* die angeforderten Informationen. * Wir suchen nach der ersten Cache-Ebene, die der CPU am nächsten liegt. * Wenn *I* gefunden werde => Cache-Treffer (*cache hit* – Erfolg); Die Suche auf dieser Ebene stoppt. * Wenn *I* nicht gefunden werde => Cache fehlgeschlagen (*cache miss*); weiterhin auf der sekundären Cache-Ebene suchen, * Wenn ein Fehler auf allen Cache-Ebenen gemeldet wurde, durchsuche ich den Hauptspeicher.   Kriterien zur Messung der Leistung eines Caches: Trefferrate (*hit rate*) = no\_hit / no\_reference to memory (der Prozentsatz erfolgreicher Treffer von den gesamten Treffern); *miss rate* = no\_miss / no\_reference to memory (Prozentsatz der fehlgeschlagenen Zugriffsversuche von der Gesamtzahl der Anforderungen), wobei *miss rate* = 1 - *hit rate*.  Die Hauptarten des Cache-Speichers:   * Hauptspeicher-Cache als Schnittstelle zwischen ihm und der CPU: Er kann sich auf einer, zwei oder drei Ebenen befinden. Dieser Cache wird von der Hardware verwaltet. * Cache zwischen Hauptspeicher und Sekundärspeicher (Festplatte) ist virtueller Speicher; Die Datenübertragung von der Festplatte zum Hauptspeicher (Verwaltung des virtuellen Speichers) wird vom Betriebssystem verwaltet. * Cache *Translation Lookaside Buffer* (TLB) der Seitentabelle, die für den Adressabgleich zwischen Hauptspeicher und virtuellem Speicher verwendet wird; * Software-verwaltete Caches: DNS-Cache (für Übereinstimmungen mit Domänennamen und IP-Adressen); Cache eines Webbrowsers (für die zuletzt aufgerufenen Seiten); Cache eines DBMS (zum Beispiel: Oracle, SQL-Server, für die zuletzt gelesenen Daten oder die neuesten entwickelten Ausführungspläne).   Im Allgemeinen hängt die Leistung eines Cache-Speichers von seiner Organisation und den Funktionsparametern ab, z. B.: der Blockplatzierungsstrategie (*block placement strategy*), der Methode zum Identifizieren einer Block (Blockidentifizierungsrichtlinie *block identification policy*) von der Cache und der Auswahlrichtlinie eines Blocks, der ersetzt werden soll, wenn im Cache kein freier Speicherplatz mehr vorhanden ist (*block replacement policy* Blockersetzungsrichtlinie), Schreiben der geänderten Daten (*cache write policy* – sofort oder wenn der Block blockiert ist). |
| 4. Dispozitivele periferice |  | 4. Die Peripheriegeräte |
| Dispozitivele periferice asigură interfaţa dintre utilizator şi sistemul de calcul sau dintre sistemul de calcul şi alte sisteme fizice. Sunt caracterizate de tipul de comportament (intrare, ieşire, stocare), partener (utilizator uman sau sistem fizic) şi performanţă.  Performanţa unui dispozitiv de intrare / ieşire (Input / Output – I/O) depinde de tipul său şi de alte componente ale sistemului (CPU, memorii, magistrale, controler şi software I/O, sistem de operare) şi este reprezentată de rata de transfer a datelor (*I/O bandwidth* = cantitatea de date transmisă şi recepţionată într-un interval de timp) şi timpul de răspuns al dispozitivului periferic (*latency*).  Categorii de dispozitive periferice:   * dispozitive de intrare: tastatură, mouse, scanner; * dispozitive de ieşire: imprimantă, monitor; * dispozitive de intrare sau ieşire: modem, placă de reţea; * dispozitive de stocare: disc (hard disk, floppy disk), bandă magnetică. |  | Die Peripheriegeräte stellen die Schnittstelle zwischen dem Benutzer und dem Computersystem oder zwischen dem Computersystem und anderen physischen Systemen bereit. Es wird durch die Art des Verhaltens (Eingabe, Ausgabe, Speicherung), des Partners (menschlicher Benutzer oder physisches System) und der Leistung charakterisiert.  Die Leistung eines Eingabe- / Ausgabegeräts (Eingabe / Ausgabe - E/A) hängt von seinem Typ und anderen Systemkomponenten (CPU, Speicher, Bus, Controller und Software-E/A, Betriebssystem) ab und wird durch die Rate von dargestellt Datenübertragung (E/A-Bandbreite *I/O bandwidth* = über einen bestimmten Zeitraum übertragene und empfangene Datenmenge) und Antwortzeit des Peripheriegeräts (*Latenz*).  Kategorien von Peripheriegeräten:   * Eingabegeräte: Tastatur, Maus, Scanner; * Ausgabegeräte: Drucker, Monitor; * Eingabe- oder Ausgabegeräte: Modem, Netzwerkkarte; * Speichergeräte: Diskette (Festplatte, Diskette), Magnetband. |
| 4.1 Magistrale – structuri de interconectare |  | 4.1 Busse - Verbindungsstrukturen |
| O magistrală este un subsistem prin care se transportă informaţie (date, instrucţiuni, semnale de control) sau energie între diferite componente ale unui SC sau între diferite SC. Spre deosebire de conexiunile punct la punct, o singură magistrală poate realiza o conexiune între două sau mai multe componente.  În sistemele de calcul moderne magistrala poate fi de tip paralel sau serial. Prin magistralele seriale se transportă informaţia ca şir de biţi (bit după bit). Magistralele paralele transportă simultan informaţie prin mai multe fire, mărindu-se astfel rata de transfer.  Acest lucru nu înseamnă că pe o magistrală paralelă viteza de transfer a informaţiei va fi neapărat mai mare decât pe una serială. Dimpotrivă, datorită costurilor mari implicate de transmisia paralelă a datelor, în ultimul timp se remarcă renunţarea la magistralele paralele şi concentrarea pe magistrale seriale care să lucreze la frecvenţe de transfer mari (de exemplu: magistrala serială S-ATA are o frecvenţă de  transfer mai mare decât magistrala paralelă IDE/ATA; similar pentru interfaţa serială USB în raport cu oricare interfaţă paralelă aflată în uz).  Un sistem de calcul include magistrale interne (locale) care fac legătura între componente interne ale sistemului (de exemplu: între CPU şi memoria internă) şi magistrale externe, pentru realizarea conexiunilor către echipamente periferice sau către alte maşini.  Sub-sistemul de magistrale al unui SC poartă numele de magistrală sistem (*system bus*). Raportat la tipul informaţiei transportate, pot fi identificate trei tipuri de magistrale într-un SC:   * *magistrale de date* (*data bus*) – transportă informaţie de tip dată sau instrucţiune. Performanţa depinde în primul rând de lăţimea de bandă (de exemplu: magistrale de 8, 16, 32, 64 biţi); * *magistrale de adrese* (*address bus*) – informaţia comunicată este adresa locaţiei de memorie pe care componenta solicitantă doreşte să o acceseze (în citire sau scriere). Lăţimea de bandă a magistralei determină capacitatea maximă de memorie adresabilă din sistem (de exemplu: sistemele de calcul cu registre pe 8 biţi cu magistrale pe 16 biţi, de unde rezultă 216 = 64K locaţii de memorie adresabile, iar sistemele PC mai noi au magistrale pe 32 biţi - 232 = 4GB locaţii); * *magistrale de control* (*control bus*) – transmit informaţii de control şi semnalizare (de exemplu: semnale de citire / scriere a memoriei, cerere de utilizare a magistralei de date acceptarea cererii de utilizare a magistralei, semnale de ceas, *reset*).   În funcţie de tipul componentelor interconectate, magistralele interne pot fi:   * magistrale CPU-memorie: asigură o comunicare rapidă directă între procesor şi memorie şi sunt de lungime redusă;      * magistrale de I/O: au în general arhitectura standardizată şi asigură o viteză ridicată în comunicarea informaţiei dintre CPU, memorie şi un controler de I/O.   Cele mai cunoscute tipuri de magistrale de I/O: ISA (*Industry Standard Architecture*), PCI (*Peripheral Component Interconnect*), PCI Express şi AGP (*Accelerated Graphics Port*). Aceste magistralele de I/O permit comunicarea cu dispozitivul periferic prin intermediul unui controler.  Fizic, un controler este o placă de extensie ataşabilă sistemului de calcul. De obicei, fiecare controler prezintă două interfeţe:     * o interfaţă de comunicare cu procesorul prin intermediul magistralei de I/O. În funcţie de tipul de magistrală folosit, această interfaţă poate fi de exemplu ISA, PCI sau AGP; * o interfaţă de comunicare cu echipamentul periferic propriu zis. Această interfaţă diferă de la un controler la altul în funcţie de echipamentul periferic care îi este ataşat. |  | Ein Bus ist ein Subsystem, über das Informationen (Daten, Anweisungen, Steuersignale) oder Energie zwischen verschiedenen Komponenten eines CS oder zwischen verschiedenen CSs transportiert werden. Im Gegensatz zu Punkt-zu-Punkt-Verbindungen kann ein einzelner Bus eine Verbindung zwischen zwei oder mehr Komponenten herstellen.  In modernen Computersystemen kann der Bus vom parallelen oder seriellen Typ sein. Die seriellen Busse übertragen die Informationen als Bitfolge (Bit für Bit). Parallele Busse übertragen Informationen gleichzeitig über mehrere Leitungen, wodurch die Übertragungsrate erhöht wird.  Dies bedeutet nicht, dass auf einem parallelen Bus die Geschwindigkeit der Informationsübertragung notwendigerweise höher als auf einem seriellen Bus ist. Im Gegenteil, aufgrund der hohen Kosten, die mit der parallelen Übertragung von Daten verbunden sind, wird in letzter Zeit der Verzicht auf parallele Busse und der Fokus auf serielle Busse, die mit hohen Übertragungsfrequenzen arbeiten, bemerkt (zum Beispiel: der serielle S-ATA-Bus hat eine Frequenz von Übertragung größer als der parallele IDE / ATA-Bus; ähnlich wie die serielle USB-Schnittstelle im Vergleich zu jeder verwendeten parallelen Schnittstelle).  Ein Computersystem umfasst interne (lokale) Busse, die die internen Komponenten des Systems (z. B. zwischen CPU und internem Speicher) und externe Busse verbinden, um Verbindungen zu Peripheriegeräten oder anderen Maschinen herzustellen.  Das Subsystem von Bussen eines CS wird als Systembus (*system bus*) bezeichnet. Abhängig von der Art der übermittelten Informationen können in einem CS drei Arten von Bussen identifiziert werden:   * *Datenbusse* (*data bus*) – Übertragen von Daten- oder Anweisungsinformationen. Die Leistung hängt in erster Linie von der Bandbreite ab (z. B. 8, 16, 32, 64-Bit-Busse). * *Adressbusse* (*address bus*) – Die übermittelte Information ist die Adresse des Speicherorts, auf den die anfordernde Komponente zugreifen möchte (beim Lesen oder Schreiben). Die Busbandbreite bestimmt die maximal adressierbare Speicherkapazität im System (zum Beispiel: 8-Bit-Registercomputersysteme mit 16-Bit-Bussen, die 216 = 64 KB adressierbare Speicherorte ergeben, und neuere PC-Systeme mit eingeschalteten Bussen 32 Bit - 232 = 4 GB Speicherorte); * *Steuerbusse* (*control bus*) – Übertragung von Steuer- und Signalisierungsinformationen (z. B. Lese- / Schreibspeichersignale, Anforderung zur Verwendung des Datenbusses, Annahme der Anforderung zur Verwendung des Busses, Taktsignale, *reset*).   Abhängig von der Art der miteinander verbundenen Komponenten können die internen Busse sein:   * CPU-Speicherbusse: sorgen für eine schnelle direkte Kommunikation zwischen Prozessor und Speicher und sind von geringer Länge; * E / A-Busse: Sie haben im Allgemeinen eine standardisierte Architektur und bieten eine hohe Geschwindigkeit bei der Kommunikation von Informationen zwischen CPU, Speicher und einem E / A-Controller.   Die beliebtesten Arten von E / A-Bussen: ISA (*Industry Standard Architecture*), PCI (*Peripheral Component Interconnect*), PCI Express und AGP (*Accelerated Graphics Port*). Diese E / A-Busse ermöglichen die Kommunikation mit dem Peripheriegerät über eine Steuerung.  Ein Controller ist eine Erweiterungskarte, die an das Computersystem angeschlossen werden kann. Normalerweise hat jeder Controller zwei Schnittstellen:   * eine Kommunikationsschnittstelle zum Prozessor über den E / A-Bus. Abhängig von der Art des verwendeten Busses kann diese Schnittstelle zum Beispiel ISA, PCI oder AGP sein; * eine Kommunikationsschnittstelle zum Peripheriegerät selbst. Diese Schnittstelle unterscheidet sich von Controller zu Controller in Abhängigkeit von den angeschlossenen Peripheriegeräten. |
|  |  |  |
| Controller  PCI-IDE  Nach CPU  PCI Bus  Controller  PCI-USB  USB Gerät  Festplatte  ISA Bus  Controller  ISA Netzwerk  Monitor  ISA Schnittstellen  PCI Schnittstellen  USB Schnittstelle  IDE Schnittstelle  AGP Schnittstelle  AGP Bus  **Figura 9.** *Magistrale de I/O (*E/A Busse*)* | | |
| 4.2. Magistrale I/O interne şi interfeţele asociate |  | 4.2. Interne E / A-Busse und die zugehörigen Schnittstellen |
| 4.2.1 Magistrala ISA |  | 4.2.1 ISA-Bus |
| A fost introdusă de IBM la începutul anilor '80, rezistând până la sfârşitul anilor '90. ISA a fost dezvoltată mai întâi pe 8 biţi, iar ulterior pe 16 biţi, operând la 8 MHz. Aceste valori au fost potrivite pentru dimensiunea magistralei sistem şi frecvenţa procesorului 286, permiţând viteze de până la 16 Megaocţeti / secundă. În timp, viteza oferită de o magistrală ISA a devenit insuficientă. Deşi, teoretic, pe o magistrală ISA se pot obţine viteze de până la 16 Megaocteţi/secundă, vitezele reale sunt mult mai mici. A fost folosită cu succes pentru toate tipurile de controlere, însă s-a bucurat de succes până la sfârşitul anilor '90 pentru conectarea la calculator mai ales a controlerelor de reţea de până la 10 Mbps, a  plăcilor de sunet şi a modemurilor.  Calculatoarele personale de astăzi, păstrează încă o relicvă a acestui tip de magistrală. Portul pentru tastatură şi mouse, iar în unele cazuri porturile seriale şi paralele, precum şi controlerul unităţii de dischetă, sunt conectate la o magistrală ISA care este cascadată la magistrala de I/O a sistemului prin intermediul magistralei PCI. |  | Es wurde von IBM in den frühen 80er Jahren eingeführt und widerstand bis in die späten 90er Jahre. Der ISA wurde zuerst mit 8 Bit und dann mit 16 Bit entwickelt und mit 8 MHz betrieben. Diese Werte waren für die Größe des Systembusses und die Prozessorfrequenz 286 geeignet und ermöglichten Geschwindigkeiten von bis zu 16 Megabit / Sekunde. Mit der Zeit wurde die von einem ISA-Bus angebotene Geschwindigkeit ungenügend. Obwohl theoretisch Geschwindigkeiten von bis zu 16 Megabyte / Sekunde auf einem ISA-Bus erreicht werden können, sind die tatsächlichen Geschwindigkeiten viel geringer. Es wurde erfolgreich für alle Arten von Controllern eingesetzt, war jedoch bis Ende der neunziger Jahre erfolgreich für den Anschluss von Netzwerkcontrollern mit bis zu 10 Mbit / s, insbesondere an den Computer Soundkarten und Modems.  Die heutigen PCs haben noch ein Relikt dieser Art von Bus. Der Tastatur- und Mausanschluss sowie in einigen Fällen der serielle und der parallele Anschluss sowie der Diskettenlaufwerk-Controller sind an einen ISA-Bus angeschlossen, der über den PCI-Bus mit dem System-E/A-Bus kaskadiert ist. |
| Imagini pentru ISA and PCI connectors  **Figura 10.** *Conectori de magistrale (*Busverbinder*): ISA, PCI, AGP* | | |
| 4.2.2 Magistrala PCI (*Peripheral Component Interconnect*) |  | **4.2.2 Der PCI-Bus (*Peripheral Component Interconnect*)** |
| A fost introdusă de Intel la sfârşitul anilor 1990. Este pe 32 de biţi şi operează la frecvenţe începând cu 33 MHz. Aceste valori permit viteze teoretice de până la 133  Megaocteţi pe secundă. Majoritatea controlerelor existente în momentul de faţă se conectează la această magistrală: controlere IDE/ATA pentru conectarea hard disk-urilor, controlere de reţea, controlere multimedia de captură, sunet, etc. Este specifică şi altor tipuri de calculatoare, nu numai calculatoarelor personale. |  | Es wurde Ende der neunziger Jahre von Intel eingeführt, ist 32-Bit-fähig und arbeitet mit Frequenzen ab 33 MHz. Diese Werte ermöglichen theoretische Geschwindigkeiten von bis zu 133 Megabyte pro Sekunde. Die meisten der derzeit vorhandenen Controller sind an diesen Bus verbunden: IDE / ATA-Controller zum Anschließen von Festplatten, Netzwerkcontrollern, Multimedia-Capture-Controllern, Sound usw. Es ist spezifisch für andere Computertypen, nicht nur für PCs. |
| 4.2.3 Magistrala AGP |  | 4.2.3 Der AGP Bus |
| S-a născut din nevoia de lăţime de bandă mai mare spre controlerul video. Aplicaţii precum jocurile cereau o lăţime de bandă pe care magistrala PCI nu o putea oferi.  De fapt numele de magistrala AGP este folosit impropriu, AGP fiind un canal de comunicaţie punct la punct. Magistrala AGP 1x este pe 32 de biţi şi lucrează la frecvenţe de 66 MHz, oferind o viteză de aproximativ 266 Megaocteţi pe secundă – dublu faţă de viteza oferită de magistrala PCI. Această lăţime de bandă este dedicată în întregime controlerului video. Magistralele AGP 8x actuale, permit atingerea unor viteze de 8 ori mai mari decât specificaţiile AGP iniţiale (2133 Megaocteţi pe secundă). |  | Es entstand aus der Notwendigkeit einer größeren Bandbreite für den Videocontroller. Anwendungen wie Spiele erforderten eine Bandbreite, die der PCI-Bus nicht bereitstellen konnte.  Tatsächlich wird der Name des AGP-Busses nicht richtig verwendet, da der AGP ein Punkt-zu-Punkt-Kommunikationskanal ist. Der AGP 1x-Bus ist 32-Bit und arbeitet mit Frequenzen von 66 MHz. Er bietet eine Geschwindigkeit von ca. 266 Megabyte pro Sekunde – doppelt so schnell wie der PCI-Bus. Diese Bandbreite ist ausschließlich dem Videocontroller vorbehalten. Die aktuellen 8x AGP-Busse ermöglichen Geschwindigkeiten, die bis zu achtmal höher sind als die ursprünglichen AGP-Spezifikationen (2133 Megabyte pro Sekunde). |
| 4.2.4 Magistrala PCI Express |  | **4.2.4 Der PCI Express Bus** |
| Magistrala PCI Express (PCIe) reprezintă a treia generație a magistralei PCI, cu o performanță și fiabilitate mai ridicate comparativ cu generațiile anterioare PCI și PCI-X. Spre deosebire de aceste versiuni anterioare, care sunt magistrale paralele, PCIe este o magistrală serială. Din cauza naturii sale seriale, magistrala PCIe are mai multe avantaje față de o magistrală paralelă: număr mai redus de pini ai circuitelor integrate, complexitate mai redusă a plăcilor de circuite imprimate și cost mai redus al acestora. Numele PCI Express reflectă atât viteza ridicată a magistralei, cât și compatibilitatea software a acesteia cu generațiile anterioare PCI și PCI-X.  Proiectanții magistralei PCIe au menținut principalele caracteristici avantajoase ale arhitecturii generațiilor anterioare de magistrale PCI. De exemplu, ea utilizează același model de comunicație ca magistralele PCI și PCI-X. Sunt păstrate aceleași spații de adrese: de memorie, de I/E și de configurație. Magistrala PCIe permite utilizarea acelorași tipuri de tranzacții ca și magistralele anterioare: citire/scriere a spațiului de memorie, citire/scriere a spațiului de I/E, citire/scriere a spațiului de configurație. Prin aceasta, se păstrează compatibilitatea cu sistemele de operare și driverele software existente, care nu necesită modificări.  Ca și magistralele PCI anterioare, magistrala PCIe permite interconexiuni între circuite integrate și interconexiuni între plăci de circuite imprimate prin conectori și plăci de extensie. Plăcile de extensie au o structură similară cu plăcile utilizate de magistralele PCI și PCI-X. O placă de bază PCIe are dimensiuni similare cu plăcile de bază ATX existente, utilizate la calculatoarele personale. Pe lângă păstrarea unor caracteristici avantajoase ale magistralelor PCI și PCI-X, magistrala PCIe introduce diferite îmbunătățiri pentru creșterea performanței și reducerea costurilor.  Magistrala PCIe utilizează o interconexiune serială punct la punct pentru comunicația între două dispozitive periferice. În primul rând, o interconexiune serială elimină dezavantajele unei magistrale paralele, în special dificultatea sincronizării între liniile multiple de date datorită nesimetriei de propagare a semnalelor. Cauza acestei nesimetrii poate fi lungimea diferită a căilor de date parcurse de diferitele semnale sau propagarea pe straturi diferite ale plăcii de circuit imprimat. Deși semnalele de date ale unei magistrale paralele sunt transmise simultan, ele pot ajunge la destinație la momente de timp diferite.  Creșterea frecvenței semnalului de ceas al unei magistrale paralele este dificilă, deoarece durata ciclului de ceas poate deveni mai mică decât durata nesimetriei de propagare a semnalelor (care poate fi de câteva nanosecunde). La o magistrală serială nu apare problema nesimetriei de propagare a semnalelor, deoarece nu există un semnal de ceas extern, informațiile de sincronizare fiind înglobate în semnalul serial transmis. În al doilea rând, o interconexiune punct la punct implică o încărcare electrică redusă a legăturii, ceea ce permite creșterea frecvenței semnalului de ceas utilizat pentru transferurile de date.  Magistrala PCIe dispune de mai multe facilități avansate. Astfel, facilitatea de calitate a serviciilor (Quality of Service–QoS) permite asigurarea unor performanțe diferențiate pentru diferite aplicații. Facilitatea de conectare și deconectare a unor module în timpul funcționării permite realizarea unor sisteme care funcționează fără întrerupere. Facilitățile de gestiune avansată a puterii + permit implementarea unor aplicații mobile cu un consum redus de putere. Facilitatea de gestiune a erorilor permite utilizarea magistralei PCIe în sistemele robuste necesare serverelor performante.  Principalele caracteristici ale magistralei PCIe sunt următoarele:   * Unifică arhitectura de I/E pentru diferite tipuri de sisteme (calculatoare de birou, calculatoare mobile, stații de lucru, servere, platforme de comunicație și sisteme înglobate) * Permite interconectarea atât a circuitelor integrate de pe placa de bază, cât și a plăcilor de extensie prin intermediul unor conectori sau cabluri. * Comunicația este bazată pe pachete, cu rată de transfer și eficiență ridicate. * Interfața este serială, permiţând reducerea numărului de pini și simplificarea conexiunilor. * Performanța este scalabilă (o interconexiune poate fi implementată cu ajutorul mai multor benzi de comunicație). * Modelul software este compatibil cu arhitectura PCI clasică (permite configurarea circuitelor PCIe, încărcarea sistemelor de operare și utilizarea driverelor software existente, fără a fi necesare modificări). * Permite o calitate diferențiată a serviciilor (QoS). * Pune la dispoziție o gestiune avansată a puterii consumate * Asigură integritatea datelor la nivelul legăturii pentru toate tipurile de tranzacții. * Permite raportarea și gestionarea avansată a erorilor pentru îmbunătățirea izolării defectelor și corectarea erorilor. * Permite conectarea și deconectarea perifericelor în timpul funcționării, fără a fi necesară utilizarea unor semnale suplimentare.   Ansamblul format din controler, indiferent de tipul de magistrală I/O internă la care este conectat acesta, împreună cu interfaţa dintre acesta şi echipamentul periferic este referit şi sub numele de *magistrală externă*. Printre cele mai cunoscute magistrale externe amintim: IDE/ATA, SCSI, USB. |  | Der PCI Express-Bus (PCIe) ist die dritte Generation des PCI-Busses mit einer höheren Leistung und Zuverlässigkeit im Vergleich zu den vorherigen PCI- und PCI-X-Generationen. Im Gegensatz zu früheren Versionen, bei denen es sich um parallele Busse handelt, PCIe einen seriellen Bus ist. Aufgrund seiner seriellen Natur hat der PCIe-Bus mehrere Vorteile gegenüber einem parallelen Bus: weniger Pins von integrierten Schaltkreisen, geringere Komplexität von Leiterplatten und geringere Kosten. Der Name PCI Express widerspiegelt sowohl die hohe Geschwindigkeit des Busses als auch die Softwarekompatibilität mit früheren PCI- und PCI-X-Generationen.  Die Entwickler des PCIe-Busses haben die wichtigsten vorteilhaften Merkmale der Architektur früherer Generationen von PCI-Bussen beibehalten. Beispielsweise verwendet es dasselbe Kommunikationsmodell wie die PCI- und PCI-X-Busse. Die gleichen Adressräume werden gespeichert: Speicher, E / A und Konfiguration. Mit dem PCIe-Bus können Sie dieselben Transaktionstypen wie mit früheren Bussen verwenden: Lese- / Schreibspeicher, Lese- / Schreib-E / A-Speicher, Lese- / Schreib-Konfigurationsspeicher. Auf diese Weise bleibt die Kompatibilität mit den vorhandenen Betriebssystemen und Softwaretreibern erhalten, für die keine Änderungen erforderlich sind.  Wie bei früheren PCI-Bussen ermöglicht der PCIe-Bus Verbindungen zwischen integrierten Schaltkreisen und Verbindungen zwischen Leiterplatten über Steckverbinder und Erweiterungskarten. Erweiterungskarten haben eine ähnliche Struktur wie PCI- und PCI-X-Busse. Ein PCIe-Motherboard hat eine ähnliche Größe wie die vorhandenen ATX-Motherboards, die auf PCs verwendet werden. Der PCIe-Bus behält nicht nur die vorteilhaften Eigenschaften von PCI- und PCI-X-Bussen bei, sondern führt auch verschiedene Verbesserungen ein, um die Leistung zu steigern und die Kosten zu senken.  Der PCIe-Bus verwendet eine serielle Punkt-zu-Punkt-Verbindung für die Kommunikation zwischen zwei Peripheriegeräten. Erstens beseitigt eine serielle Verbindung die Nachteile eines parallelen Busses, insbesondere die Schwierigkeit der Synchronisation zwischen mehreren Datenleitungen aufgrund der Asymmetrie der Signalausbreitung. Die Ursache für diese Asymmetrie kann die unterschiedliche Länge der Datenpfade sein, die von den unterschiedlichen Signalen zurückgelegt werden, oder die Ausbreitung auf unterschiedlichen Schichten der Leiterplatte. Die Datensignale eines Parallelbusses werden zwar gleichzeitig übertragen, können aber zu unterschiedlichen Zeiten das Ziel erreichen.  Das Erhöhen der Frequenz des Taktsignals eines Parallelbusses ist schwierig, da die Dauer des Taktzyklus kürzer werden kann als die Dauer der Signalausbreitungsasymmetrie (die einige Nanosekunden betragen kann). In einem seriellen Bus gibt es kein Problem der Asymmetrie der Signalausbreitung, da es kein externes Taktsignal gibt, wobei die Synchronisationsinformation in dem übertragenen seriellen Signal enthalten ist. Zweitens impliziert eine Punkt-zu-Punkt-Verbindung eine verringerte elektrische Ladung der Verbindung, wodurch die Frequenz des für Datenübertragungen verwendeten Taktsignals erhöht werden kann.  Der PCIe-Bus verfügt über mehrere erweiterte Funktionen. Somit ermöglicht die *Quality of Service* (QoS) -Funktion differenzierte Leistungen für verschiedene Anwendungen. Die einfache Verbindung und Trennung von Modulen während des Betriebs ermöglicht die Realisierung von Systemen, die ohne Unterbrechung arbeiten. Erweiterte Energieverwaltungsfunktionen ermöglichen die Bereitstellung mobiler Anwendungen mit geringem Energieverbrauch. Die Fehlermanagementfunktion ermöglicht die Verwendung des PCIe-Busses in den robusten Systemen, die für Hochleistungsserver erforderlich sind.  Die Hauptmerkmale des PCIe-Busses sind:   * Vereinheitlicht die E / A-Architektur für verschiedene Systemtypen (Bürocomputer, mobile Computer, Workstations, Server, Kommunikationsplattformen und eingebettete Systeme) * Ermöglicht die Verbindung sowohl der integrierten Schaltkreise auf der Hauptplatine als auch der Erweiterungsplatinen über Steckverbinder oder Kabel. * Die Kommunikation erfolgt paketbasiert mit hoher Übertragungsrate und Effizienz. * Die Schnittstelle ist seriell, wodurch die Anzahl der Pins verringert und die Verbindungen vereinfacht werden. * Die Performance ist skalierbar (eine Verschaltung kann mit Hilfe mehrerer Kommunikationsbänder realisiert werden). * Das Softwaremodell ist mit der klassischen PCI-Architektur kompatibel (es ermöglicht die Konfiguration von PCIe-Schaltkreisen, das Laden von Betriebssystemen und die Verwendung vorhandener Softwaretreiber, ohne dass Änderungen erforderlich sind). * Ermöglicht eine differenzierte Servicequalität (QoS). * Bietet erweiterte Verwaltung des Stromverbrauchs * Gewährleistet die Datenintegrität auf Verbindungsebene für alle Arten von Transaktionen. * Ermöglicht erweiterte Fehlerberichte und -verwaltung, um die Fehlerisolierung und -korrektur zu verbessern. * Ermöglicht das Anschließen und Trennen von Peripheriegeräten während des Betriebs, ohne dass zusätzliche Signale erforderlich sind.   Die Baugruppe bestehend aus der Steuerung, unabhängig vom Typ des internen E / A-Busses, an den sie angeschlossen ist, und der Schnittstelle zwischen ihr und den Peripheriegeräten wird auch als *externer Bus* bezeichnet. Zu den beliebtesten externen Bussen zählen: IDE / ATA, SCSI, USB. |
| 4.3 Magistrale I/O externe şi interfeţele asociate |  | 4.3 E/A Busse und zugehörige Schnittstellen |
| 4.3.1 IDE / ATA (*Integrated Drive Electronics / Advanced Technology Attachment*) |  | 4.3.1 IDE / ATA (*Integrated Drive Electronics / Advanced Technology Attachment*) |
| Interfaţa IDE sau ATA a fost folosită de la mijlocul anilor '80 în vederea conectării perifericelor de stocare cum ar fi hard disk-urile şi unităţile optice. Primele specificaţii ATA foloseau procesorul (aşa numitul *programmed input/output mode* sau mod PIO) pentru a transfera informaţia octet cu octet de pe dispozitivul periferic de stocare în memoria internă prin intermediul registrelor procesorului. Acest mod de lucru permitea obţinerea de viteze relativ mici de până la 16.7 Megaocteţi pe secundă şi, mai grav, reţineau procesorul ocupat în timpul acestui transfer. Această viteză a crescut însă odată cu folosirea DMA (*Direct Memory Access*), permiţându-se astfel copierea informaţiei în blocuri de pe dispozitivul de stocare direct în memoria internă fără a mai implica procesorul în acest transfer.  Vitezele actuale ale interfeţelor ATA 133 se apropie cel puţin teoretic de limitele magistralei PCI de 132 Megaocteţi pe secundă pentru transferul datelor (să nu uităm că informaţia citită de pe dispozitivul de stocare trebuie să treacă prin magistrala PCI via controler).  Specificaţiile ATA descriu şi modul de adresare a dispozitivului de stocare. Ultimele specificaţii prevăd o adresare pe 48 de biţi, lucru care permite folosirea teoretică a dispozitivelor de stocare la capacităţi de până la 32 de Teraocteţi.  Pe un controler IDE/ATA se pot conecta doar două dispozitive de stocare, unul operând în mod master şi celălalt în mod slave. Majoritatea calculatoarelor personale, datorită faptului că sunt dotate cu două controlere IDE (numite primary şi secondary IDE), pot folosi la un moment dat doar 4 dispozitive periferice IDE. Dacă se doreşte folosirea unui număr mai mare de periferice IDE/ATA, este necesară instalarea unui controler IDE/ATA suplimentar, de obicei PCI. |  | Die IDE- oder ATA-Schnittstelle wird seit Mitte der 1980er Jahre zum Anschließen von Speicherperipheriegeräten wie Festplatten und optischen Laufwerken verwendet. Die ersten ATA-Spezifikationen verwendeten den Prozessor (den sogenannten *programmed input/output mode* oder den PIO-Modus), um die Byte-Byte-Informationen vom Peripheriespeichergerät über die Prozessorregister in den internen Speicher zu übertragen. Diese Betriebsart ermöglichte relativ niedrige Geschwindigkeiten von bis zu 16,7 Megabyte pro Sekunde und hielt den Prozessor während dieser Übertragung schlimmer noch beschäftigt. Diese Geschwindigkeit hat sich jedoch mit der Verwendung von *Direct Memory Access* (DMA) erhöht, wodurch die Blockinformationen direkt von der Speichervorrichtung in den internen Speicher kopiert werden können, ohne dass der Prozessor an dieser Übertragung beteiligt ist.  Die aktuellen Geschwindigkeiten der ATA 133-Schnittstellen liegen theoretisch mindestens nahe an den PCI-Busgrenzen von 132 Megabit pro Sekunde für die Datenübertragung (denken Sie daran, dass die vom Speichergerät gelesenen Informationen über den Controller über den PCI-Bus übertragen werden müssen).  Die ATA-Spezifikationen beschreiben auch, wie das Speichergerät adressiert wird. Die neuesten Spezifikationen umfassen eine 48-Bit-Adresse, die die theoretische Verwendung von Speichergeräten mit einer Kapazität von bis zu 32 Terabyte ermöglicht.  An einen IDE / ATA-Controller können nur zwei Speichergeräte angeschlossen werden, eines im Master-Modus und das andere im Slave-Modus. Die meisten PCs können, da sie mit zwei IDE-Controllern (primäre und sekundäre IDE) ausgestattet sind, nur 4 IDE-Peripheriegeräte gleichzeitig verwenden. Wenn Sie mehr IDE / ATA-Peripheriegeräte verwenden möchten, müssen Sie einen zusätzlichen IDE / ATA-Controller installieren, in der Regel PCI. |
| 4.3.2 SCSI (*Small Computer System Interface*) |  | 4.3.2 SCSI (*Small Computer System Interface*) |
| Acest tip de magistrală s-a născut la mijlocul anilor '80, fiind cel mai des folosită pentru conectarea dispozitivelor de stocare cum ar fi hard disk-uri, unităţi optice, unităţi de bandă magnetică. Poate fi folosită şi pentru conectarea altor tipuri de periferice (scannere, imprimante).  Poate fi folosită atât ca magistrală internă, cât şi ca magistrală externă. În calculatoarele personale, a fost folosită doar ca magistrală externă, comunicând cu procesorul doar prin intermediul unui controler ISA sau PCI prin magistrala internă corespunzătoare tipului de controler. Funcţional, magistrala SCSI operează la frecvenţe cuprinse între 5 MHz (conform primelor specificaţii SCSI) şi 80 de MHz, dimensiunea magistralei fiind de 8 sau 16 biţi. O magistrală SCSI pe 16 biţi operând la 80 MHz poate atinge viteze de până la 320 Megaocteţi pe secundă.  Fiecărui dispozitiv periferic conectat la o magistrală SCSI i se asociază un identificator *SCSIid*. Numărul de biţi pe care se reprezintă acest identificator implică şi numărul de dispozitive care se pot conecta pe aceeaşi magistrală. Astfel, spre deosebire de IDE care permitea doar două dispozitive periferice per controler (magistrală), o magistrală SCSI poate suporta până la 8 sau 16 echipamente periferice. Un alt avantaj al controlerelor SCSI este că suportă *hot-swapping* (*hot-plugging*) – schimbarea în timpul mersului calculatorului a echipamentelor periferice conectate. Interfaţa serială comunică serial cu un dispozitiv periferic, transferul de date făcându-se pe principiul serial, bit cu bit. Cele mai des întâlnite periferice seriale sunt mouse-urile, modemurile şi terminalele virtuale. |  | Dieser Bustyp wurde Mitte der 1980er Jahre geboren und wird am häufigsten zum Anschließen von Speichergeräten wie Festplatten, optischen Laufwerken und Magnetbandlaufwerken verwendet. Es kann auch zum Anschließen anderer Peripheriegeräte (Scanner, Drucker) verwendet werden.  Es kann sowohl als interner Bus als auch als externer Bus verwendet werden. In Personalcomputern wurde er nur als externer Bus verwendet und kommunizierte mit dem Prozessor nur über einen ISA- oder PCI-Controller über den internen Bus, der dem Controllertyp entspricht. Funktionell arbeitet der SCSI-Bus mit Frequenzen zwischen 5 MHz (gemäß der ersten SCSI-Spezifikation) und 80 MHz, wobei die Busgröße 8 oder 16 Bit beträgt. Ein 16-Bit-SCSI-Bus mit 80 MHz kann Geschwindigkeiten von bis zu 320 Megabit pro Sekunde erreichen.  Jedem an einen SCSI-Bus angeschlossenen Peripheriegerät wird eine *SCSIid*-Kennung zugewiesen. Die Anzahl der Bits, die durch diesen Bezeichner dargestellt werden, impliziert auch die Anzahl der Geräte, die an denselben Bus angeschlossen werden können. Im Gegensatz zu IDE, die nur zwei Peripheriegeräte pro Controller (Bus) zulässt, kann ein SCSI-Bus somit bis zu 8 oder 16 Peripheriegeräte unterstützen. Ein weiterer Vorteil von SCSI-Controllern besteht darin, dass sie *hot-swapping* (*hot-Plugging*) unterstützen – das Umschalten angeschlossener Peripheriegeräte während des Betriebs des Computers. Die serielle Schnittstelle kommuniziert seriell mit einem Peripheriegerät, wobei die Datenübertragung nach dem seriellen Prinzip Bit für Bit erfolgt. Die gebräuchlichsten seriellen Peripheriegeräte sind Mäuse, Modems und virtuelle Terminals. |
| 4.3.3 Interfaţa serială |  | 4.3.3 Serielle Schnittstelle |
| Ea comunică serial cu un dispozitiv periferic, transferul de date făcându-se pe principiul serial, bit cu bit. Cele mai des întâlnite periferice seriale sunt mouse-urile, modemurile şi terminalele virtuale.  Controlerul serial la calculatoarele personale de azi este pe cale de dispariţie, majoritatea perifericelor care se conectau la calculator pe această interfaţă conectându-se în prezent prin USB. Acolo unde este încă prezent, controlerul serial este legat de magistrala ISA sau PCI. Viteza maximă atinsă pe un port serial este foarte mică, 128000 biţi /secundă ≈ 16 kiloocteţi / secundă. |  | Sie kommuniziert seriell mit einem Peripheriegerät, wobei die Datenübertragung nach dem seriellen Prinzip Bit für Bit erfolgt. Die gebräuchlichsten seriellen Peripheriegeräte sind Mäuse, Modems und virtuelle Terminals.  Der serielle Controller heutiger PCs ist vom Aussterben bedroht. Die meisten Peripheriegeräte, die über diese Schnittstelle an den Computer angeschlossen sind, sind derzeit über USB angeschlossen. Wo es noch vorhanden ist, wird der serielle Controller an den ISA- oder PCI-Bus angeschlossen. Die maximale Geschwindigkeit, die an einem seriellen Port erreicht wird, ist sehr niedrig: 128000 Bit / Sekunde ≈ 16 Kilobyte / Sekunde. |
| 4.3.4 Interfaţa paralelă |  | 4.3.4 Parallele Schnittstelle |
| Este şi ea o relicvă în calculatoarele moderne, încet renunţându-se la ea şi datorită nevoii de a reduce costurile unui sistem de calcul şi mai ales datorită numărului mic de periferice care mai folosesc acest tip de interfaţă. Principiul de comunicare pe o asemenea interfaţă este bineînţeles cel paralel – mai mulţi biţi sunt transferaţi în acelaşi timp pe mai multe fire fizice.  În prezent, singurele echipamente întâlnite care folosesc această interfaţă sunt imprimantele. În trecut, interfaţa paralelă a fost folosită şi pentru conectarea altor periferice cum ar fi camerele video sau scannerele. Producătorii de asemenea echipamente periferice au renunţat la interfaţa paralelă în favoarea celei USB. Ca şi în cazul controlerului serial, unde mai este prezent, controlerul paralel este legat de magistrala ISA sau PCI. |  | Es ist auch ein Relikt in modernen Computern, das langsam aufgibt und aufgrund der Notwendigkeit, die Kosten eines Computersystems zu senken, und insbesondere aufgrund der geringen Anzahl von Peripheriegeräten, die diese Art von Schnittstelle verwenden. Das Kommunikationsprinzip auf einer solchen Schnittstelle ist natürlich das parallele – viele Bits werden gleichzeitig auf mehreren physischen Drähten übertragen.  Derzeit sind die einzigen Geräte, die diese Schnittstelle verwenden, Drucker. In der Vergangenheit wurde die parallele Schnittstelle auch zum Anschluss anderer Peripheriegeräte wie Camcorder oder Scanner verwendet. Hersteller solcher Peripheriegeräte haben auch die parallele Schnittstelle zugunsten von USB aufgegeben. Wie beim seriellen Controller, wo er noch vorhanden ist, ist der parallele Controller mit dem ISA- oder PCI-Bus verbunden. |
| 4.3.5 USB (Universal Serial Bus) |  | 4.3.5 USB (Universal Serial Bus) |
| Interfaţa USB s-a născut din necesitatea unui mecanism universal de conectare a echipamentelor periferice la calculator, indiferent de tipul acestora. Rata de transfer merge de la 1.5 Mbit/secundă (USB 1.0) la 10 Gbit/secundă (USB 3.1 Gen 2), suficientă pentru conectarea oricărui dispozitiv extern cu calculatorul, mai puţin a unui monitor. Principiul de comunicare cu acestea este serial, interfaţa USB folosind pentru date doar două fire. Pe un singur + USB se pot conecta până la 127 de echipamente USB. Fiecare echipament primind un USB id reprezentat pe 7 biţi (de fapt un id este consumat de controlerul USB însuşi).  Specificaţiile USB permit adăugarea echipamentelor periferice USB sau înlocuirea acestora chiar în timpul funcţionării calculatorului. |  | Die USB-Schnittstelle entstand aus der Notwendigkeit eines universellen Mechanismus zum Anschließen von Peripheriegeräten an den Computer, unabhängig von ihrem Typ. Die Übertragungsrate reicht von 1,5 Mbit / s (USB 1.0) bis 10 Gbit / s (USB 3.1 Gen 2), um alle externen Geräte mit Ausnahme eines Monitors an den Computer anzuschließen. Das Prinzip der Kommunikation mit diesen ist seriell, wobei die USB-Schnittstelle für Daten nur zwei Drähte verwendet. Bis zu 127 USB-Geräte können an einen einzelnen USB-Controller angeschlossen werden. Jedes Gerät erhält eine 7-Bit-USB-ID (tatsächlich wird eine ID vom USB-Controller selbst verwendet).  USB-Spezifikationen ermöglichen das Hinzufügen oder Ersetzen von USB-Peripheriegeräten, auch wenn der Computer ausgeführt wird. |
| 5. Arhitectura microprocesoarelor x86 (IA-32) |  | 5. Architektur von x86-Mikroprozessoren (IA-32) |
| 5.1 Structura microprocesorului |  | 5.1 Aufbau des Mikroprozessors |
| Microprocesorul x86 este format din două componente principale:   * **EU** (*Executive Unit*) – execută instrucţiunile maşină prin intermediul componentei ALU. * **BIU** (*Bus Interface Unit*) – pregăteşte execuţia fiecărei instrucţiuni maşină. Citeşte o instrucţiune din memorie, o decodifică şi calculează adresa din memorie a unui eventual operand. Configuraţia rezultată este depusă într-o zonă tampon cu dimensiunea de 15 octeţi, de unde va fi preluată de EU.   EU si BIU lucrează în paralel – în timp ce EU execută instrucţiunea curentă, BIU pregăteşte instrucţiunea următoare. Cele două acţiuni sunt sincronizate – cea care termină prima aşteaptă după cealaltă. |  | Der x86-Mikroprozessor besteht aus zwei Hauptkomponenten:   * **EU** (*Executive Unit*) – führt die Maschinenanweisungen über die ALU-Komponente aus. * **BIU** (*Bus Interface Unit*) – bereitet die Ausführung jeder MaschinenAnweisung vor. Liest einen Anweisung aus dem Speicher, decodiert ihn und berechnet die Speicheradresse eines möglichen Operanden. Die resultierende Konfiguration wird in einer Pufferzone mit einer Größe von 15 Byte gespeichert und von der EU übernommen.   EU und BIU arbeiten parallel – während die EU die aktuelle Anweisung ausführt, bereitet die BIU die nächste Anweisung vor. Diesen Aktionen sind synchronisiert – die eine, die die erste beendet, wartet nach der anderen. |
| **Figura 11.** *Arhitectura microprocesorului 8086 (*Architektur von 8086-Mikroprozessor*)* | | |
| 5.1.1 Regiştrii generali ai EU |  | 5.1.1 EU-Allgemeine Register |
| 1. **Registrul EAX** (*Extended AX*) este registrul acumulator. El este folosit de către majoritatea instrucţiunilor ca unul dintre operanzi; 2. **Registrul EBX** – registru general; 3. **Registrul ECX** – registru de numărare (registru contor) pentru instrucţiunile care au nevoie de indicaţii numerice. 4. **Registrul EDX** – registru de date. Împreună cu EAX se foloseşte în calculele ale căror rezultate depăşesc un dublucuvânt (32 biţi).   Un „cuvânt binar” (*binary word*) este o reprezentare a unui număr într-un sistem de numeraţie binar pe un anumit număr de biţi dat. „Dimensiunea cuvântului” se referă la numărul de biţi procesaţi de CPU-ul unui SC la un moment dat (de regulă, 32 de biţi sau 64 de biţi). Dimensiunea sau lăţimea de bandă a magistralelor de date sau de adrese, dimensiunea instrucţiunilor etc. sunt de regulă multipli ai acestui număr.  Pentru a complica lucrurile, pentru aşa-numita *backwards compatibility* (compatibilitate retroactivă), Microsoft Windows API defineşte un WORD ca fiind un TIP DE DATE pe 16 biţi, un DWORD ca fiind un TIP DE DATE pe 32 de biţi şi un QWORD pe 64 de biţi, indiferent de procesor.   1. **Regiştrii ESP şi EBP** sunt regiştri destinaţi lucrului cu stiva. O stivă se defineşte ca fiind o zonă de memorie în care se pot depune succesiv valori, extragerea lor ulterioară făcându-se în ordinea inversă depunerii. Registrul ESP (*Stack Pointer*) punctează spre elementul ultim introdus în stivă (elementul din vârful stivei). Registrul EBP (*Base pointer*) punctează spre primul element introdus în stivă (indică baza stivei). 2. **Regiştrii EDI şi ESI** (*Destination Index* şi *Source Index*) sunt regiştrii de index utilizaţi de obicei pentru accesarea elementelor din şiruri de octeţi sau de cuvinte.   Fiecare dintre regiştrii EAX, EBX, ECX, EDX, ESP, EBP, EDI, ESI au capacitatea de 32 biţi. Fiecare dintre ei poate fi privit în acelaşi timp ca fiind format prin concatenarea (alipirea) a doi (sub)regiştri de câte 16 biți. Subregistrul superior, care conţine cei mai semnificativi 16 biţi ai registrului de 32 biţi din care face parte, nu are denumire și nu este disponibil separat. Subregistrul inferior poate însă fi accesat individual, având astfel regiștrii de 16 biți AX, BX, CX, DX, SP, BP, DI, SI. Dintre aceștia, regiștrii AX, BX, CX, şi DX sunt, fiecare la rândul lor, formați din câte doi alți subregiștri a câte 8 biți. Există astfel regiştrii AH, BH, CH, DH, conținând cei 8 biți superiori (partea HIGH a regiștrilor AX, BX, CX și DX), respectiv AL, BL, CL, DL, conținând cei 8 biți inferiori (partea LOW). |  | 1. Das **EAX-Register** (*Extended AX*) ist das Akumulatorregister. Es wird von den meisten Anweisungen als einer der Operanden verwendet; 2. **EBX-Register** - Allgemeines Register; 3. **ECX-Register** – Zählregister für die Anweisungen, die numerische Informationen benötigen. 4. **EDX-Register** – Datenregister. Zusammen mit EAX wird es in Berechnungen verwendet, deren Ergebnisse ein Doppelwort (32 Bit) überschreiten.   Ein „binäres Wort“ ist eine Darstellung einer Zahl in einem binären Nummerierungssystem für eine gegebene Anzahl von Bits. „Wortgröße“ bezieht sich auf die Anzahl von Bits, die von der CPU eines CS zu einer gegebenen Zeit verarbeitet werden (normalerweise 32 Bits oder 64 Bits). Die Größe oder Bandbreite der Daten- oder Adressbusse, die Größe der Anweisungen usw. in der Regel sind Vielfaches dieser Zahl.  Um die Abwärtskompatibilität (*backwards compatibility*) zu gewährleisten, definiert die Microsoft Windows-API ein WORD als 16-Bit-Datentyp, ein DWORD als 32-Bit-Datentyp und ein 64-Bit-QWORD. Bits, unabhängig vom Prozessor.   1. **ESP- und EBP-Register** sind Register, die für die Stapelverarbeitung vorgesehen sind. Ein Stapel ist definiert als ein Speicherbereich, in dem Werte nacheinander abgelegt werden können, wobei ihre anschließende Entnahme in umgekehrter Reihenfolge der Ablage erfolgt. Das Register ESP (*Stack Pointer*) zeigt auf das zuletzt in den Stapel eingefügte Element (das Element oben im Stapel). Das EBP-Register (*Base Pointer*) zeigt auf das erste Element, das in den Stapel eingegeben wurde (gibt die Basis des Stapels an). 2. **Die EDI und ESI** **Register** (*Destination Index* und *Source Index*) sind Indexregister, die üblicherweise für den Zugriff auf Elemente in Byte- oder Wortfolgen verwendet werden.   Jedes der EAX-, EBX-, ECX-, EDX-, ESP-, EBP-, EDI- und ESI-Register hat eine Kapazität von 32 Bit. Jedes von ihnen kann gleichzeitig als durch Verketten (Anhängen) von zwei (Unter-) Registern mit jeweils 16 Bits gebildet angesehen werden. Das obere Unterregister, das die höchstwertigen 16 Bits des 32-Bit-Registers enthält, zu dem es gehört, hat keinen Namen und ist nicht separat verfügbar. Auf den unteren Teilbereich kann jedoch individuell zugegriffen werden, so dass die 16-Bit-Register AX, BX, CX, DX, SP, BP, DI, SI vorliegen. Davon bestehen die Register AX, BX, CX und DX jeweils aus zwei weiteren 8-Bit-Unterregistern. Es gibt also Register AH, BH, CH, DH, die die 8 oberen Bits enthalten (der HIGH-Teil der Register AX, BX, CX und DX) bzw. AL, BL, CL, DL, die die 8 unteren Bits enthalten (der LOW-Teil). |
| **Figura 12.** *Structura registrului EAX (*Struktur des EAX-Registers*)* | | |
| 5.1.2 Flagurile |  | 5.1.2 Flaggen |
| Un *flag* este un indicator reprezentat pe un bit. O configuraţie a *registrului de flag*-uri indică un rezumat sintetic al execuţiei fiecărei instrucţiuni. Pentru x86 registrul EFLAGS (the *status* register) are 32 biţi dintre care sunt folosiţi uzual numai 9. |  | Ein Flag ist ein Indikator, der auf einem Bit dargestellt wird. Eine *Flagregisterkonfiguration* gibt eine Zusammenfassung der Ausführung jeder Anweisung an. Für x86 hat das EFLAGS-Register (das *Statusregister*) 32 Bits, von denen nur 9 gemeinsam verwendet werden. |
| |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | **31** | **30** | **…** | **12** | **11** | **10** | **9** | **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** | | x | x | … | x | OF | DF | IF | TF | SF | ZF | x | AF | x | PF | x | CF | | | |
| **CF** (*Carry Flag*) este *flag*-ul de transport. Are valoarea 1 în cazul în care în cadrul ultimei operații efectuate (UOE) s‑a efectuat transport în afara domeniului de reprezentare a rezultatului şi valoarea 0 în caz contrar. De exemplu, pentru: |  | CF (*Carry Flag*) ist die Transportflagge. Es hat den Wert 1, wenn in der zuletzt ausgeführten Operation (ZAO) außerhalb des Darstellungsfeldes des Ergebnisses transportiert wurde und den Wert 0, ansonsten. Zum Beispiel für: |
| |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 1001 0011 | + |  | 146 | + |  | 93h | + |  | -109 | + | | 0111 0011 |  |  | 115 |  |  | 73h |  |  | 115 |  | | **1**0000 0110 |  |  | **0**262 |  |  | **1**06h |  |  | 06 |  | | | |
| rezultă un transport de cifră semnificativă şi valoarea 1 este depusă automat în CF.  **Flagul CF semnalează depăşirea în cazul interpretării FĂRĂ SEMN.**  **PF** (*Parity Flag*) – Valoarea lui se stabileşte astfel încât împreună cu numărul de biţi de 1 din octetul cel mai puțin semnificativ al reprezentării rezultatului UOE să rezulte un număr impar de cifre 1.  **AF** (*Auxiliary Flag*) indică valoarea transportului de la bitul 3 la bitul 4 al rezultatului UOE. De exemplu, în adunarea de mai sus bitul de transport este 0.  **ZF** (*Zero Flag*) primeşte valoarea 1 dacă rezultatul UOE este egal cu zero şi valoarea 0 la rezultat diferit de zero.  **SF** (*Sign Flag*) primeşte valoarea 1 dacă rezultatul UOE este un număr strict negativ şi valoarea 0 în caz contrar.  **TF** (*Trap Flag*) este un *flag* de depanare. Dacă are valoarea 1, atunci maşina se opreşte după fiecare instrucţiune.  **IF** (*Interrupt Flag*) este *flag* de întrerupere. Asupra acestui *flag* vom reveni.    **DF** (*Direction Flag*) – pentru operarea asupra şirurilor de octeţi sau de cuvinte. Dacă are valoarea 0, atunci deplasarea în şir se face de la început spre sfârşit, iar dacă are valoarea 1 este vorba de deplasări de la sfârşit spre început.  **OF** (*Overflow Flag*) este *flag* pentru depăşire în cazul operaţiilor asupra numerelor **CU SEMN**. Dacă rezultatul ultimei instrucţiuni în interpretarea CU SEMN a operanzilor nu a încăput în spaţiul rezervat operanzilor (intervalul de reprezentare admisibil), atunci acest *flag* va avea valoarea 1, altfel va avea valoarea 0. |  | es wird eine signifikante Zahl transportiert und der Wert 1 wird automatisch im CF abgelegt.  **Das CF-Flag signalisiert die Überwindung bei Interpretation OHNE ZEICHEN.**  **PF** (*Parity Flag*) – Sein Wert wird so eingestellt, dass sich zusammen mit der Anzahl der Bits von 1 des niedrigstwertigen Bytes der Darstellung des ZAO-Ergebnisses eine ungerade Anzahl von Ziffern 1 ergibt.  **AF** (*Auxiliary Flag*) zeigt den Transportwert von Bit 3 bis Bit 4 des ZAO-Ergebnisses an. In der obigen Versammlung ist der Transportbit beispielsweise 0.  **ZF** (*Zero Flag*) erhält den Wert 1, wenn das Ergebnis der ZAO gleich Null ist und der Wert 0 am Ergebnis von Null abweicht.  **SF** (*Sign Flag*) erhält den Wert 1, wenn das ZAO-Ergebnis eine streng negative Zahl ist, und den Wert 0, wenn dies nicht der Fall ist.  **TF** (*Trap Flag*) ist ein Fehlerbehebungsflag. Wenn es auf 1 gesetzt ist, stoppt die Maschine nach jeder Anweisung.  **IF** (*Interrupt Flag*) ist das Interrupt-Flag. Auf diese Flagge werden wir zurückkommen.  **DF** (*Direction Flag*) – für den Betrieb an Byte- oder Wortketten. Wenn es den Wert 0 hat, wird die Bewegung in der Zeichenfolge vom Anfang bis zum Ende ausgeführt, und wenn es den Wert 1 hat, handelt es sich um die Bewegungen vom Ende bis zum Anfang.  **OF** (*Overflow Flag*) ist das Flag, das bei Operationen mit **Vorzeichenzahle** überschritten werden soll. Wenn das Ergebnis der letzten Anweisung in der Vorzeichenzahle-Interpretation der Operanden nicht in den für die Operanden reservierten Bereich (zulässiger Darstellungsbereich) passte, hat dieses Flag den Wert 1, andernfalls den Wert 0. |
| 5.1.3 Regiştrii de adresă şi calculul de adresă |  | 5.1.3 Adressregister und Adressberechnung |
| *Definiţie*  *Adresa unei locaţii* este numărul de **octeţi** consecutivi dintre începutul memoriei RAM şi începutul locaţiei respective.  O succesiune continuă de locaţii de memorie, menite să deservească scopuri similare în timpul execuţiei unui program, formează un *segment*. În consecinţă, un segment reprezintă o diviziune logică a memoriei unui program, caracterizată prin *adresa de bază* (început), *limita* (dimensiunea) şi *tipul* acesteia. Atât adresa de bază cât şi dimensiunea unui segment au valori reprezentate pe 32 biţi.  În familia procesoarelor bazate pe arhitectura 8086, termenul **segment** are două înţelesuri:  1. Un bloc de memorie de dimensiune discretă, numit segment fizic. Numărul de octeţi dintr-un segment de memorie fizică este de 64K pentru procesoarele pe 16 biţi, respectiv 4 GB pentru procesoarele pe 32 de biţi.  2. Un bloc de memorie de dimensiune variabilă, numit segment logic, ocupat de codul sau datele unui program.  Vom numi *offset* sau *deplasament* adresa unei locaţii faţă de începutul unui segment, sau, cu alte cuvinte, numărul de octeţi aflaţi între începutul segmentului şi locaţia în cauză. Un *offset* se consideră valid dacă şi numai dacă valoarea sa numerică, pe 32 biţi, nu depăşeşte limita segmentului la care se raportează.  Vom numi *specificare de adresă* (sau *adresă logică*) o pereche formată dintr-un *selector de segment* şi un *offset*. Un ***selector de segment*** este o valoare numerică de 16 biţi care identifică (indică/selectează) în mod unic segmentul accesat şi caracteristicile acestuia. În scriere hexazecimală o adresă se exprimă sub forma: |  | *Definition*  *Die Adresse eines Speicherorts* ist die Anzahl aufeinanderfolgender **Bytes** zwischen dem Beginn des RAM und dem Beginn dieses Speicherorts.  Eine fortlaufende Folge von Speicherplätzen, die während der Programmausführung ähnlichen Zwecken dienen sollen, bildet ein *Segment*. Folglich stellt ein Segment eine logische Aufteilung des Programmspeichers dar, die durch seine *Basisadresse* (Start), *Grenze* (Größe) und seinen *Typ* gekennzeichnet ist. Sowohl die Basisadresse als auch die Größe eines Segments haben Werte, die auf 32 Bits dargestellt sind.  In der Familie der Prozessoren, die auf der 8086-Architektur basieren, hat der Begriff **Segment** zwei Bedeutungen:  1. Ein Speicherblock mit diskreter Größe, der als *physikalisches Segment* bezeichnet wird. Die Anzahl der Bytes in einem physischen Speichersegment beträgt 64 KB für 16-Bit-Prozessoren bzw. (beziehungsweise) 4 GB für 32-Bit-Prozessoren.  2. Ein Speicherblock mit variabler Größe, der als *logisches Segment* bezeichnet wird und mit dem Code oder den Daten eines Programms belegt ist.  Wir nennen *Offset* die Adresse eines Ortes gegenüber dem Beginn eines Segments, oder mit anderen Worten die Anzahl der Bytes zwischen dem Beginn des Segments und dem betreffenden Ort. Ein *Offset* gilt nur dann als gültig, wenn sein numerischer Wert auf 32 Bit die Segmentgrenze, auf die er sich bezieht, nicht überschreitet.  Wir werden die *Adressspezifikation* (oder *logische Adresse*) ein Paar nennen, das aus einem *Segmentselektor* und einem *Offset* besteht. Ein ***Segmentselektor*** ist ein numerischer 16-Bit-Wert, der das Segment, auf das zugegriffen wird, und seine Eigenschaften eindeutig identifiziert (anzeigt / auswählt). In hexadezimaler Schreibweise wird eine Adresse ausgedrückt als: |
| ***s*3*s*2*s*1*s*0 : *o*7*o*6*o*5*o*4*o*3*o*2*o*1*o*0** | | |
| În acest caz, selectorul *s*3*s*2*s*1*s*0 indică accesarea unui segment a cărui adresă de bază este de forma *b*7*b*6*b*5*b*4*b*3*b*2*b*1*b*0 şi având o limită *l*7*l*6*l*5*l*4*l*3*l*2*l*1*l*0. Baza şi limita sunt determinate de către procesor în urma aplicării mecanismului de segmentare.  Pentru a fi permis accesul către locaţia specificată, este necesar să fie îndeplinită condiţia: |  | In diesem Fall zeigt der Selektor s3s2s1s0 den Zugriff auf ein Segment an, dessen Basisadresse die Form *b*7*b*6*b*5*b*4*b*3*b*2*b*1*b*0 hat und dessen Grenze *l*7*l*6*l*5*l*4*l*3*l*2*l*1*l*0 ist. Die Basis und die Grenze werden vom Prozessor nach Anwendung des Segmentierungsmechanismus bestimmt.  Um Zugriff auf den angegebenen Ort zu erhalten, muss die Bedingung erfüllt sein: |
| ***o*7*o*6*o*5*o*4*o*3*o*2*o*1*o*0 ≤  *l*7*l*6*l*5*l*4*l*3*l*2*l*1*l*0** | | |
| Determinarea *adresei de segmentare* din specificarea de adresă se face conform formulei: |  | Die Ermittlung der Segmentierungsadresse aus der Adressangabe erfolgt nach der Formel: |
| ***a*7*a*6*a*5*a*4*a*3*a*2*a*1*a*0****:= *b*7*b*6*b*5*b*4*b*3*b*2*b*1*b*0** **+ *o*7*o*6*o*5*o*4*o*3*o*2*o*1*o*0** | | |
| unde *a*7*a*6*a*5*a*4*a*3*a*2*a*1*a*0 este adresa calculată (scrisă în hexazecimal). Adresa rezultată din calculul de mai sus, poartă numele de *adresă liniară.*  O specificare de adresă mai poartă şi numele de adresă FAR (îndepărtată). Atunci când o adresă se precizează doar prin *offset*, spunem ca este o adresă NEAR (apropiată), pentru că se subînţelege că operăm în acelaşi segment.  Un exemplu concret de specificare de adresă este: |  | dabei ist *a*7*a*6*a*5*a*4*a*3*a*2*a*1*a*0 die berechnete Adresse (hexadezimal geschrieben). Die aus der obigen Berechnung resultierende Adresse wird als lineare Adresse bezeichnet.  Eine Adressspezifikation trägt auch den FAR (*remote*) -Adressennamen. Wenn eine Adresse nur durch einen *Offset* angegeben wird, sagen wir, dass es sich um eine NEAR (schließen) -Adresse handelt, da davon ausgegangen wird, dass wir im selben Segment arbeiten.  Ein konkretes Beispiel für eine Adressangabe ist: |
| **8:1000h** | | |
| Pentru a calcula adresa liniară care corespunde acestei specificări, procesorul va proceda după cum urmează:   1. Verifică dacă segmentul care corespunde valorii de selector 8 a fost definit de către sistemul de operare. Dacă un astfel de segment nu a fost definit, se blochează accesul; 2. Extrage adresa de bază (B) şi limita acestui segment (L), de exemplu, ca rezultat am putea avea B = 2000h şi L = 4000h; 3. Verifică dacă *offset*-ul depăşeşte limita segmentului: 1000h > 4000h? În caz de depăşire, accesul ar fi fost blocat; 4. Adună *offset*-ul cu B, obţinând în cazul nostru adresa liniară 3000h (1000h + 2000h). Acest calcul este efectuat de către componenta **ADR** din **BIU.**   Acest mecanism de adresare poartă numele de *segmentare*, vorbind astfel despre *modelul de adresare* *segmentată.*  În cazul în care segmentele încep la adresa 0 şi au dimensiunea maximă posibilă (4GB), orice *offset* este automat valid şi segmentarea nu contribuie efectiv în calculul adreselor. Astfel, având *b*7*b*6*b*5*b*4*b*3*b*2*b*1*b*0 = 00000000, calculul de adresă pentru adresa logică *s*3*s*2*s*1*s*0:*o*7*o*6*o*5*o*4*o*3*o*2*o*1*o*0 va rezulta în adresa liniară: |  | Um die dieser Spezifikation entsprechende lineare Adresse zu berechnen, geht der Prozessor wie folgt vor:   1. Überprüft, ob das Segment, das dem Auswahlwert 8 entspricht, vom Betriebssystem definiert wurde. Wenn ein solches Segment nicht definiert wurde, wird der Zugriff gesperrt. 2. Extrahiert die Basisadresse (B) und die Grenze dieses Segments (L). Als Ergebnis könnten B = 2000h und L = 4000h sein. 3. Überprüft, ob der *Offset* die Segmentgrenze überschreitet: 1000h> 4000h? Im Falle einer Überwindung wäre der Zugang gesperrt worden; 4. Sammelt den *Offset* mit B und erhaltet in unserem Fall die lineare Adresse 3000h (1000h + 2000h). Diese Berechnung wird von der ADR-Komponente der BIU durchgeführt.   Dieser Adressierungsmechanismus trägt den *Segmentierungsnamen* und spricht somit vom *segmentierten Adressierungsmodell*.  Wenn die Segmente bei Adresse 0 beginnen und die maximal mögliche Größe (4 GB) haben, ist jeder *Offset* automatisch gültig und die Segmentierung trägt nicht effektiv zur Berechnung der Adressen bei. Wenn also *b*7*b*6*b*5*b*4*b*3*b*2*b*1*b*0 = 00000000 ist, ergibt die Adressberechnung für die logische Adresse *s*3*s*2*s*1*s*0:*o*7*o*6*o*5*o*4*o*3*o*2*o*1*o*0 die lineare Adresse: |
| ***a*7*a*6*a*5*a*4*a*3*a*2*a*1*a*0 := 00000000 + *o*7*o*6*o*5*o*4*o*3*o*2*o*1*o*0**  ***a*7*a*6*a*5*a*4*a*3*a*2*a*1*a*0 := *o*7*o*6*o*5*o*4*o*3*o*2*o*1*o*0** | | |
| Acest mod particular de utilizare a segmentării, folosit de către majoritatea sistemelor de operare moderne poartă numele de *model de memorie flat*.  Procesoarele x86 suportă şi un mecanism de control al accesului la memorie. Acesta se numeşte *paginare* şi este independent de adresarea segmentată. Paginarea implică împărţirea memoriei *virtuale* în *pagini,* care sunt asociate (se spune că sunt translatate) memoriei fizice disponibile.  Configurarea şi controlul mecanismelor de segmentare şi de paginare sunt sarcina sistemului de operare. Dintre cele două, doar segmentarea intervine în specificarea de adrese, paginarea fiind complet transparentă din perspectiva programelor de utilizator („transparent” înseamnă că programele nu „văd” nimic cu privire la acest mecanism).  Atât calculul de adrese cât şi folosirea mecanismelor de segmentare şi paginare sunt influenţate de *modul de execuţie* al procesorului, procesoarele x86 suportând următoarele moduri de execuţie mai importante:   1. *mod real*, pe 16 biţi (folosind cuvânt de memorie de 16 biţi şi având memoria limitată la 1 MB); 2. ***mod protejat pe 16 sau 32 biţi*, caracterizat prin folosirea paginării şi segmentării**; 3. *mod virtual 8086*, permite rularea programelor de tip mod real alături de cele de mod protejat; 4. *long mode*, pe 64 sau 32 biţi, unde paginarea este obligatorie în timp ce segmentarea este dezactivată.   În cadrul cursului ne vom concentra asupra arhitecturii şi comportamentului procesoarelor x86 în modul protejat pe 32 de biţi.  Arhitectura x86 permite folosirea a patru tipuri de segmente cu roluri diferite:   * *segment de cod*, care conţine instrucţiuni maşină; * *segment de date*, care conţine date asupra cărora se acţionează în conformitate cu instrucţiunile; * *segment de stivă*; * *segment suplimentar* *de date* (extrasegment).   Fiecare program este compus din unul sau mai multe segmente, de unul sau mai multe dintre tipurile de mai sus. În fiecare moment al execuţiei este activ cel mult câte un segment din fiecare tip. Regiştrii **CS** (*Code Segment*), **DS** (*Data Segment*), **SS** (*Stack Segment*), **ES** (*Extra Segment*) din **BIU** reţin valorile selectorilor segmentelor active, corespunzător fiecărui tip. Deci regiştrii CS, DS, SS şi ES determină adresele de început şi dimensiunile segmentelor active: de cod, de date, de stivă şi suplimentar. Regiştrii **FS** şi **GS** pot reţine selectori indicând către segmente suplimentare, fără însă a avea roluri predeterminate. Datorită utilizării lor, CS, DS, SS, ES, FS şi GS poartă denumirea de *regiştri de segment* (sau *regiştri selectori*). Registrul **EIP** (care oferă şi posibilitatea accesării cuvântului său inferior prin subregistrul **IP**) conţine *offset*-ul instrucţiunii curente în cadrul segmentului de cod curent, el fiind manipulat exclusiv de către **BIU**.  Noţiunile asociate adresării sunt fundamentale pentru înţelegerea funcţionării procesoarelor x86 şi programării în limbaj de asamblare. Pentru aceasta, în Tabelul 1 le recapitulăm pe scurt în vederea clarificării: |  | Diese spezielle Art der Verwendung der Segmentierung, die von den meisten modernen Betriebssystemen verwendet wird, wird als *flaches Speichermodell* bezeichnet.  Die x86-Prozessoren unterstützen auch einen Mechanismus zur Speicherzugriffskontrolle. Dies wird als *Paging* bezeichnet und ist unabhängig von der segmentierten Adressierung. Bei einer Seite wird der *virtuelle* Speicher in *Seiten* aufgeteilt, die sind dem verfügbaren physischen Speicher zugeordnet (sie sollen übersetzbar sein).  Das Konfigurieren und Steuern der Segmentierungs- und Paging-Mechanismen ist Aufgabe des Betriebssystems. Von den beiden ist nur die Segmentierung bei der Angabe von Adressen beteiligt, wobei die Seite aus Sicht von Benutzerprogrammen vollständig transparent ist („transparent“ bedeutet, dass die Programme nichts über diesen Mechanismus „sehen“).  Sowohl die Adressen werden berechnet als auch die Segmentierungs- und Paging-Mechanismen werden vom dem *Ausführung Modus* im Prozessor beeinflusst, wobei die x86-Verarbeitung direkt wichtigere Ausführungsmodi unterstützt:   1. 16-Bit-*Real-Modus* (unter Verwendung eines 16-Bit-Speicherworts und einer begrenzten Sprache von 1 MB); 2. ***16-Bit-geschützter 32-Bit-Modus*, gekennzeichnet durch die Verwendung von Paging und Segmentierung;** 3. *8086 virtuellen Modus* ermöglicht die Anpassung von Programmen, um die realen Modi des berühmten geschützten Modus zu ändern; 4. Der *lange Modus* 64 sieht aus wie 32 Bit und ist schädlich, wenn die Segmentierung deaktiviert ist.   Während des Kurses konzentrieren wir uns auf die Architektur und das Verhalten von x86-Prozessoren im 32-Bit-geschützten Modus.  Die x86-Architektur ermöglicht die Verwendung von vier Segmenttypen mit unterschiedlichen Rollen:   * *Codesegment*, das MaschinenAnweisungen enthält; * *Datensegment*, das Daten enthält, auf die es gemäß den Anweisungen einwirkt; * *Stapelsegment*; * *zusätzliches Datensegment* (Extrasegment).   Jedes Programm besteht aus einem oder mehreren Segmenten, einem oder mehreren der oben genannten Typen. Zu jedem Ausführungszeitpunkt ist höchstens ein Segment jedes Typs aktiv. Die Register **CS** (*Code Segment*), **DS** (*Data Segment*), **SS** (*Stack Segment*), **ES** (*Extra Segment*) in der **BIU** behalten die Werte der Selektoren der aktiven Segmenten bei, die jedem Typ entsprechen. Die Register CS, DS, SS und ES bestimmen also die Startadressen und die Dimensionen der aktiven Segmente: Code, Daten, Stapel und Additional. Die **FS**- und **GS**-Register können Selektoren behalten, die auf zusätzliche Segmente zeigen, ohne vorbestimmte Rollen zu haben. CS, DS, SS, ES, FS und GS werden aufgrund ihrer Verwendung als *Segmentregister* (oder *Selektorregister*) bezeichnet. Das **EIP**-Register (das auch die Möglichkeit bietet, über das **IP**-Unterregister auf sein unteres Wort zuzugreifen) enthält den *Offset* des aktuellen Anweisung innerhalb des aktuellen Codesegments, der ausschließlich von der **BIU** verarbeitet wird.  Die mit der Adressierung verbundenen Begriffe sind von grundlegender Bedeutung, um die Funktionsweise von x86-Prozessoren und die Programmierung in Assemblersprache zu verstehen. Zu diesem Zweck fassen wir sie in Tabelle 1 zusammen, um Folgendes zu verdeutlichen: |
| **Tabelul 1.** *Adresarea în cadrul microprocesoarelor x8086 (*Adressierung innerhalb von x8086-Mikroprozessoren*)*   |  |  |  | | --- | --- | --- | | **Noţiune (*Bedeutung*)** | **Reprezentare (*Darstellung*)** | **Descriere (*Beschreibung*)** | | Specificare de adresă, adresă logică, adresă FAR (*Adressangabe, logische Adresse, FAR-Adresse*) | Selector16:offset32  (*Selektor*16:*offset*32) | Defineşte complet atât segmentul cât şi deplasamentul în cadrul acestuia  (*Es definiert sowohl das Segment als auch die Verschiebung innerhalb des Segments vollständig*) | | Selector (*Selektor*) | 16 biţi | Identifică unul dintre segmentele disponibile. Ca valoare numerică acesta codifică poziţia descriptorului de segment selectat în cadrul unei tabele de descriptori.  (*Identifiziert eines der verfügbaren Segmente. Als numerischer Wert codiert er die Position des ausgewählten Segmentdeskriptors in einer Deskriptortabelle.*) | | *Offset*, adresă NEAR (*NEAR-Adresse*) | *Offset*32 | Defineşte doar componenta de *offset* (considerând segmentul cunoscut ori folosirea modelului de memorie *flat*)  [*Definiert nur die* Offset*-Komponente (unter Berücksichtigung des bekannten Segments oder unter Verwendung des Flat Memory-Modells)*] | | Adresă liniară  (adresă de segmentare) [*Lineare Adresse*  (*Segmentierungsadresse*)] | 32 biţi | Început segment + *offset*, reprezintă rezultatul calculului de segmentare  (*Startsegment + Offset ist das Ergebnis der Segmentierungsberechnung*) | | Adresă fizică efectivă (*Effektive physikalische Adresse*) | Cel puţin 32 biţi (*Mindestens 32 Bit*) | Rezultatul final al segmentării plus, eventual, paginării. Adresa finală obţinută de către BIU, indicând în memoria fizică (hardware)  [*Das Endergebnis der Segmentierung plus möglicherweise die Paginierung. Die endgültige Adresse, die von der BIU erhalten wird und die im physischen Speicher (Hardware) angezeigt wird*] |  Diferențe cheie între paginare și segmentare  1. Diferența de bază între paginare și segmentare este că o pagină are întotdeauna **o dimensiune fixă ​​a blocului**, în timp ce un segment este de **mărime variabilă** . 2. Pagingul poate duce la **fragmentarea internă** deoarece pagina este de dimensiune fixă ​​a blocului, dar se poate întâmpla ca procesul să nu obțină întreaga dimensiune a blocului care va genera fragmentul intern din memorie. Segmentarea poate duce la **fragmentarea externă,** deoarece memoria este umplută cu blocurile cu dimensiuni variabile. 3. În paginare, utilizatorul oferă doar un **întreg întreg** ca adresă care este divizată de hardware într-un **număr de pagină și Offset** . Pe de altă parte, în segmentare utilizatorul specifică adresa în două cantități, adică **numărul segmentului și offsetul** . 4. Mărimea paginii este decisă sau specificată de **hardware** . Pe celelalte mâini, dimensiunea segmentului este specificată de **utilizator** . 5. În paginare, tabela de **pagini** hartă **adresa logică la adresa fizică** și conține adresa de bază a fiecărei pagini stocate în cadrele spațiului de memorie fizică. Cu toate acestea, în segmentare, **tabelul segmentului** hartă **adresa logică la adresa fizică** și conține numărul de segment și offset (limita segmentului). | | |
| 5.2 Reprezentarea instrucţiunilor maşină |  | 5.2 Darstellung von Maschinenanweisungen |
| O instrucţiune maşină x86 reprezintă o secvenţă de 1 până la 15 octeţi, care prin valorile lor specifică o operaţie de executat, operanzii asupra cărora va fi aplicată, precum şi modificatori suplimentari care controlează modul în care aceasta va fi executată.  O instrucţiune maşină x86 are maximum doi operanzi. Pentru cele mai multe dintre instrucţiuni, cei doi operanzi poartă numele de *sursă*, respectiv *destinaţie*. Dintre cei doi operanzi, maximum unul se poate afla în memoria RAM. Celălalt se află fie într-un registru al **EU**, fie este o constantă întreagă. Astfel, o instrucţiune are forma: |  | Ein x86-MaschinenAnweisung stellt eine Sequenz von 1 bis 15 Byte dar, die durch ihre Werte eine auszuführende Operation, die Operanden, auf die er angewendet wird, sowie zusätzliche Modifikatoren angibt, die steuern, wie er ausgeführt wird.  Ein x86-MaschinenAnweisung hat maximal zwei Operanden. Bei den meisten Anweisungen werden die beiden Operanden als *Quelle* bzw. *Ziel* bezeichnet. Maximal einer der beiden Operanden kann sich im RAM Speicher befinden. Die andere befindet sich entweder in einem EU-Register oder ist eine Ganzzahlkonstante. Eine Anweisung hat also die Form: |
| **Nume\_instrucţiune** *destinaţie*, *sursă* (**Anweisung\_name** *Ziel, Quelle*) | | |
| Formatul intern al unei instrucţiuni este variabil, el putând ocupa între 1 şi 15 octeţi, având următoarea formă generală de reprezentare (*Instructions byte-codes from OllyDbg*): |  | Das interne Format einer Anweisung ist variabel, es kann zwischen 1 und 15 Byte belegen und die folgende allgemeine Darstellungsform haben (*Instructions byte-codes from OllyDbg*): |
| *[prefixe] + cod + [ModR/M] + [SIB] + [deplasament] + [imediat]*  ([Präfixe] + Code + [ModR / M] + [SIB] + [Offset] + [Sofort]) | | |
| *Prefixele* controlează modul în care o instrucţiune se execută. Acestea sunt opţionale (0 până la maximum 4) şi ocupă câte un octet fiecare. De exemplu, acestea pot solicita execuţia repetată (în buclă) a instrucţiunii curente sau pot bloca magistrala de adrese pe parcursul execuţiei pentru a nu permite accesul concurent la operanzi şi rezultate.  Operaţia care se va efectua este identificată prin intermediul a 1 sau 2 octeţi de *cod* (*opcode*), aceştia fiind singurii octeţi obligatoriu prezenţi, indiferent de instrucţiune. |  | Präfixe steuern, wie eine Anweisung ausgeführt wird. Diese sind optional (0 bis maximal 4) und belegen jeweils ein Byte. Beispielsweise können sie eine wiederholte Ausführung (Schleife) des aktuellen Anweisung erfordern oder den Adressbus während der Ausführung blockieren, um einen gleichzeitigen Zugriff auf Operanden und Ergebnisse zu verhindern.  Die auszuführende Operation wird durch 1 oder 2 Byte *Code* (*Opcode*) identifiziert, wobei dies die einzigen vorhandenen Bytes sind, unabhängig von der Anweisung. |
| The x86 instruction format  **Figura 13.** *Prefixele şi formatele instrucțiunilor microprocesorului 8086 (*Präfixe und Formate für 8086-Mikroprozessoranweisungen*)* | | |
| Deşi Figura 13 pare a indica faptul că instrucţiunile pot avea lungimi de până la 16 octeţi, în realitate microprocesorul x86 nu permite instrucţiuni cu lungimi mai mari de 15 octeţi.  Octetul *ModR/M* (mod registru/memorie) specifică pentru unele dintre instrucţiuni natura şi locul operanzilor (registru, memorie, constantă întreagă etc.). Acesta permite specificarea fie a unui registru, fie a unei locaţii de memorie a cărei adresă este exprimată prin intermediul unui *offset*.  Pentru cazuri mai complexe de adresare decât cele codificabile direct prin ModR/M, combinarea acestuia cu octetul SIB permite următoarea formulă generală de definire a unui *offset*: |  | Obwohl in Abbildung 13 anscheinend angegeben ist, dass die Anweisungen eine Länge von bis zu 16 Byte haben können, lässt der x86-Mikroprozessor in Wirklichkeit keine Anweisungen mit einer Länge von mehr als 15 Byte zu.  Das Byte ModR / M (Register- / Memory Modus) spezifiziert für einige Anweisungen die Art und den Ort der Operanden (Register, Speicher, Ganzzahlkonstante usw.). Es kann entweder ein Register oder ein Speicherort angegeben werden, dessen Adresse durch einen *Offset* ausgedrückt wird.  Bei komplexeren Adressierungsfällen als denen, die direkt von ModR/M codiert werden, ermöglicht die Kombination mit dem SIB-Byte die folgende allgemeine Formel zur Definition eines *Offsets*: |
| **[bază] + [index × scală] + [constantă]**  **([*Basis*] + [*Index* × *Skala*] + [*Konstante*])** | | |
| unde pentru bază şi index vor fi folosite valorile a doi regiştri iar scală este 1, 2, 4 sau 8. Regiştrii permişi ca bază sau / şi index sunt: EAX, EBX, ECX, EDX, EBP, ESI, EDI. Registrul ESP este disponibil ca bază însă nu poate fi folosit cu rol de index.  Majoritatea instrucţiunilor folosesc pentru reprezentare fie numai câmpul de cod, fie cod urmat de ModR/M.  *Deplasament* (*displacement*) apare în cazul unor forme de adresare particulare (operanzi din memorie) şi urmează direct după ModR/M sau SIB, când SIB este prezent. Acest câmp poate fi codificat fie pe octet fie pe dublucuvânt (32 biţi).  Ca şi consecinţă a imposibilităţii prezenţei mai multor câmpuri de ModR/M, SIB şi deplasament într-o instrucţiune, arhitectura nu permite codificarea a două adrese de memorie în aceeaşi instrucţiune.  *Valoare* *imediată* oferă posibilitatea definirii unui operand ca fiind o constantă numerică pe 1, 2 sau 4 octeţi. Când este prezent, acest câmp apare întotdeauna la sfârşitul instrucţiunii. |  | Wobei für die Basis und den Index die Werte von zwei Registern verwendet werden und die Skala 1, 2, 4 oder 8 ist. Die zulässigen Register als Basis und / oder Index sind: EAX, EBX, ECX, EDX, EBP, ESI, EDI. Das ESP-Register ist als Basis verfügbar, kann jedoch nicht als Index verwendet werden.  Die meisten Anweisungen verwenden zur Darstellung entweder nur das Codefeld oder den Code, gefolgt von ModR / M.  Die *Verschiebung* (*displacement*) tritt bei bestimmten Adressierungsformen (Operanden aus dem Speicher) auf und folgt unmittelbar nach ModR/M oder SIB, wenn SIB vorhanden ist. Dieses Feld kann entweder byteweise oder durch Doppelwort (32 Bit) codiert werden.  Infolge der Unmöglichkeit, dass mehrere Felder von ModR/M, SIB und Verschiebung in einem Befehl vorhanden sind, erlaubt die Architektur nicht die Codierung von zwei Speicheradressen in dieselbe Anweisung.  Der *Sofortwert* bietet die Möglichkeit, einen Operanden als numerische Konstante auf 1, 2 oder 4 Bytes zu definieren. Wenn vorhanden, erscheint dieses Feld immer am Ende der Anweisung. |
| 5.3 Adrese FAR şi NEAR |  | 5.3 FAR- und NEAR-Adressen |
| Pentru a adresa o locaţie din memoria RAM sunt necesare două valori: una care să indice segmentul, alta care să indice *offset*-ul în cadrul segmentului. Pentru a simplifica referirea la memorie, microprocesorul derivă, în lipsa unei alte specificări, adresa segmentului din unul dintre regiştrii de segment CS, DS, SS sau ES. Alegerea implicită a unui registru de segment se face după nişte reguli proprii instrucţiunii folosite.  Prin definiţie, o adresă în care se specifică doar *offset*-ul, urmând ca segmentul să fie preluat implicit dintr-un registru de segment poartă numele de *adresă NEAR* (adresă apropiată). O adresă NEAR se află întotdeauna în interiorul unuia din cele patru segmente active.  O adresă în care programatorul indică explicit un selector de segment poartă numele de *adresă FAR* (adresă îndepărtată). O adresă FAR este deci o SPECIFICARE COMPLETĂ DE ADRESĂ şi ea se poate exprima în trei moduri:   * *s*3*s*2*s*1*s*0 : specificare\_offset unde *s*3*s*2*s*1*s*0 este o constantă; * registru\_segment:specificare\_offset, registru segment fiind CS, DS, SS, ES, FS sau GS; * FAR [variabilă], unde variabilă este de tip QWORD şi conţine cei 6 octeţi constituind adresa FAR.   Formatul intern al unei adrese FAR este: la adresa mai mică se află offsetul, iar la adresa mai mare cu 4 (cuvântul care urmează după dublucuvântul curent) se află cuvântul care conţine selectorul care indică segmentul.  Reprezentarea adreselor respectă principiul reprezentării little-endian: partea cea mai puţin semnificativă are adresa cea mai mică, iar partea cea mai semnificativă are adresa cea mai mare. |  | Um eine Position im RAM zu adressieren, sind zwei Werte erforderlich: einer zur Angabe des Segments und ein anderer zur Angabe des *Offsets* innerhalb des Segments. Um die Speicherreferenz zu vereinfachen, leitet der Mikroprozessor, sofern nicht anders angegeben, die Segmentadresse aus einem der Segmentregister CS, DS, SS oder ES ab. Die Standardauswahl eines Segmentregisters erfolgt nach bestimmten Regeln, die für die verwendete Anweisung spezifisch sind.  Per Definition trägt eine Adresse, bei der nur der *Offset* angegeben ist, gefolgt von dem Segment, das standardmäßig aus einem Segmentregister entnommen wird, den *NEAR-Adressennamen.* Eine NEAR-Adresse befindet sich immer in einem der vier aktiven Segmente.  Eine Adresse, bei der der Programmierer ausdrücklich einen Segmentwähler angibt, wird als *FAR-Adresse* (ferne Adresse) bezeichnet. Eine FAR-Adresse ist daher eine vollständige Adressangabe und kann auf drei Arten ausgedrückt werden:   * *s*3*s*2*s*1*s*0: Offset-Angabe, wobei *s*3*s*2*s*1*s*0 eine Konstante ist; * Segmentregister:Offset-Spezifikation, wobei das Segmentregister CS, DS, SS, ES, FS oder GS ist; * FAR [Variable], wobei die Variable vom Typ QWORD ist und die 6 Bytes enthält, die die FAR-Adresse bilden.   Das interne Format einer FAR-Adresse lautet: Bei der niedrigeren Adresse ist der *Offset* und bei der größeren Adresse mit 4 (dem Wort nach dem aktuellen Doppelwort) das Wort, das den Selektor enthält, der das Segment angibt.  Die Adressendarstellung respektiert das Prinzip der Little-Endian-Darstellung: Der niedrigstwertige Teil hat die kleinste Adresse und der höchstwertige Teil hat die höchste Adresse. |
| 5.4 Calculul *offset*-ului unui operand. Moduri de adresare |  | 5.4 Berechnung des *Offsets* eines Operanden. Adressierungsarten |
| În cadrul unei instrucţiuni există 3 moduri de a specifica un operand pe care aceasta îl solicită:   * *modul registru*, dacă pe post de operand se află un registru al maşinii; * *modul imediat*, atunci când în instrucţiune se află chiar valoarea operandului (nu adresa lui şi nici un registru în care să fie conţinut); * *modul adresare la memorie*, dacă operandul se află efectiv undeva în memorie. În acest caz, adresa *offset*-ului lui se calculează după următoarea formulă: |  | Innerhalb einer Anweisung gibt es drei Möglichkeiten, einen angeforderten Operanden anzugeben:   * der *Registermodus*, wenn sich ein Maschinenregister auf dem Operanden befindet; * der *unmittelbare Modus*, wenn sich der Wert des Operanden in die Anweisung befindet (nicht seine Adresse und kein Register, in dem er enthalten ist); * *Speicheradressierungsmodus*, wenn sich der Operand tatsächlich irgendwo im Speicher befindet. In diesem Fall wird die Adresse seines *Offsets* nach folgender Formel berechnet: |
| ***adresa\_offset* = [ bază ] + [ index × scală ] + [ constanta]**  **(Offset\_Adresse = [*Basis*] + [*Index* × *Skala*] + [*Konstante*])** | | |
| Deci *adresa\_offset* se obţine din următoarele (maxim) patru elemente:   * conţinutul unuia dintre regiştrii EAX, EBX, ECX, EDX, EBP, ESI, EDI sau ESP ca bază; * conţinutul unuia dintre regiştrii EAX, EBX, ECX, EDX, EBP, ESI sau EDI drept index; * factor numeric (scală) pentru a înmulţi valoarea registrului index cu 1, 2, 4 sau 8 * valoarea unei constante numerice, pe octet sau dublucuvânt.   De aici rezultă următoarele moduri de adresare a memoriei:   * *directă*, atunci când apare numai *constanta*; * *bazată*, dacă în calcul apare unul dintre regiştrii bază; * *scalat-indexată*, dacă în calcul apare unul dintre regiştrii index;   Cele trei moduri de adresare a memoriei pot fi combinate. De exemplu, poate să apară adresare directă bazată, adresare bazată şi scalat-indexată etc.  La instrucţiunile de salt mai apare şi un alt tip de adresare numit adresare *relativă*.  *Adresa relativă* indică poziţia următoarei instrucţiuni de executat, în raport cu poziţia curentă. Poziţia este indicată prin numărul de octeţi de cod peste care se va sări. Arhitectura x86 permite atât adrese relative scurte (*SHORT Addresses*), reprezentate pe octet şi având valori între -128 şi 127, cât şi adrese relative apropiate (NEAR Addresses), pe dublucuvânt cu valori între -2147483648 şi 2147483647. |  | Daher wird die *Offset\_Adresse* aus den folgenden (maximal) vier Elementen erhalten:   * der Inhalt eines der EAX-, EBX-, ECX-, EDX-, EBP-, ESI-, EDI- oder ESP-Register als Basis; * der Inhalt eines der Register EAX, EBX, ECX, EDX, EBP, ESI oder EDI als Index; * numerischer Faktor (Skala), um den Wert des Indexregisters mit 1, 2, 4 oder 8 zu multiplizieren; * der Wert einer numerischen Konstante pro Byte oder Doppelwort.   Ab hier ergeben sich folgende Möglichkeiten zur Adressierung des Speichers:   * *direkt*, wenn nur die *Konstante* erscheint; * *basierend*, wenn eines der Basisregister in der Berechnung erscheint; * *skaliert-indiziert*, wenn eines der Indexregister in der Berechnung erscheint;   Die drei Adressierungsarten des Speichers können kombiniert werden. Beispielsweise können eine direktbasierte Adressierung, eine basierte Adressierung und eine skalierte- *indizierte* Adressierung usw.  Auftreten eine andere Art von Adresse, die als *relative* Adressierung bezeichnet wird, erscheint auch in den SprungAnweisungen. Die *relative Adresse* gibt die Position der folgenden auszuführenden Anweisung relativ zur aktuellen Position an. Die Position wird durch die Anzahl der zu überspringenden Code-Bytes angegeben. Die x86-Architektur ermöglicht sowohl kurze relative Adressen (*SHORT Addresses*), die durch Byte dargestellt werden und Werte zwischen -128 und 127 aufweisen, als auch nahe relative Adressen (NEAR Addresses) in einem Duplikat mit Werten zwischen -2147483648 und 2147483647. |