Principles of Computer Architecture
Processors

Lecture 1

prof.dr. Dalius Mažeika
Dalius.Mazeika@vgtu.lt
http://dma.vgtu.lt
Room No. L424

Lectures and evaluation

Scope of the course is 3 ECTS (80 hours)

- Theory – 30 hours (15 lectures)
- Practice – 15 hours (7 lectures)

Structure of the final grade

- Practice coefficient is 0.3
- Exam coefficient is 0.7

Literature

- W. Stalling. Computer organizations and architecture. 8th Edition, Pearson, 2010.
- A. S. Tanenbaum, D. J. Wetherall. **Computer Networks**. Prentice Hall. 2010.
- J. F. Kurose, K. W. Ross. **Computer Networking. A Top Down Approach**. Addison Wesley. 2010.
- C. M. Kozierok. **The TCP/IP Guide**. 2005
- V.Urbanavičius. **Kompiuteriai ir jų architektūra**. VGTU. 2007. (*www.ebooks.vgtu.lt*)

Content of the theory course

- **Computer architecture**
  - Processors (word, pipes, ISA, ILP, SMP, threads, cores)
  - RAMs (SRAM, DRAM, SDRAM, DDR, CAS, RAS, latency)
  - Cache(L1, L2, L3, Direct mapped, Fully Associative, Set Associative)
  - HDD, SDD, RAID, LUN, file systems
  - File storages (DAS, NAS, SAN, protocols)

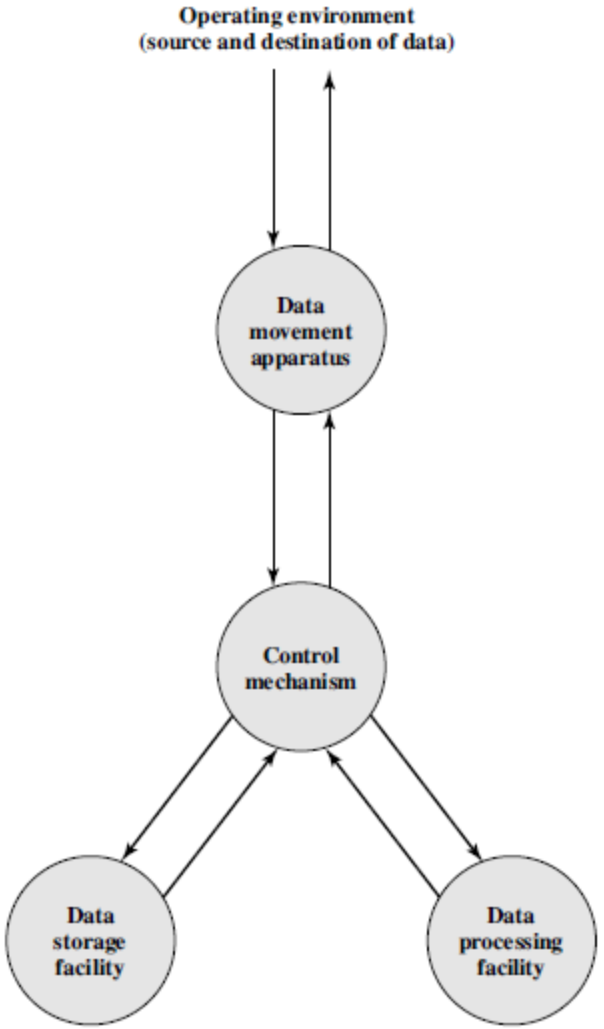
Content of the theory course

- **Computer networks**
  - Network topology, components, main features
  - OSI ir TCP/IP models
  - L2 layer - LAN technologies, IEEE 802.x standard
  - L3 layer, IP protocol (IP v4, IPv6)
  - L4 layer, TCP and UDP protocols
  - WAN technologies
  - Wireless LAN (IEEE 802.11x)
  - Routing and protocols (BGP, RIP, OSPF)
  - Application layer (DNS, DHCP, FTP, ICMP, SMTP, etc.)

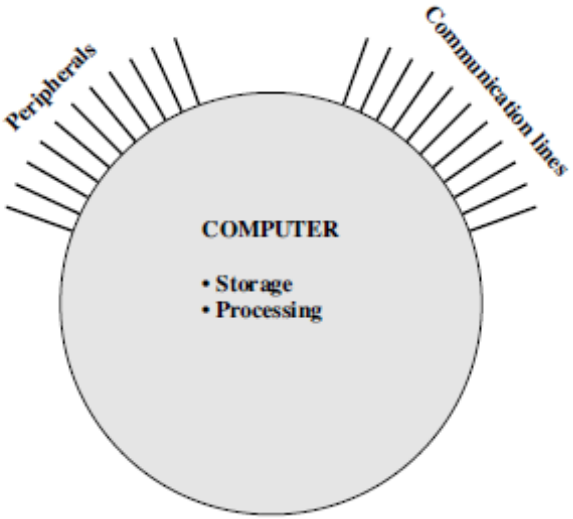
Functional view of the computer

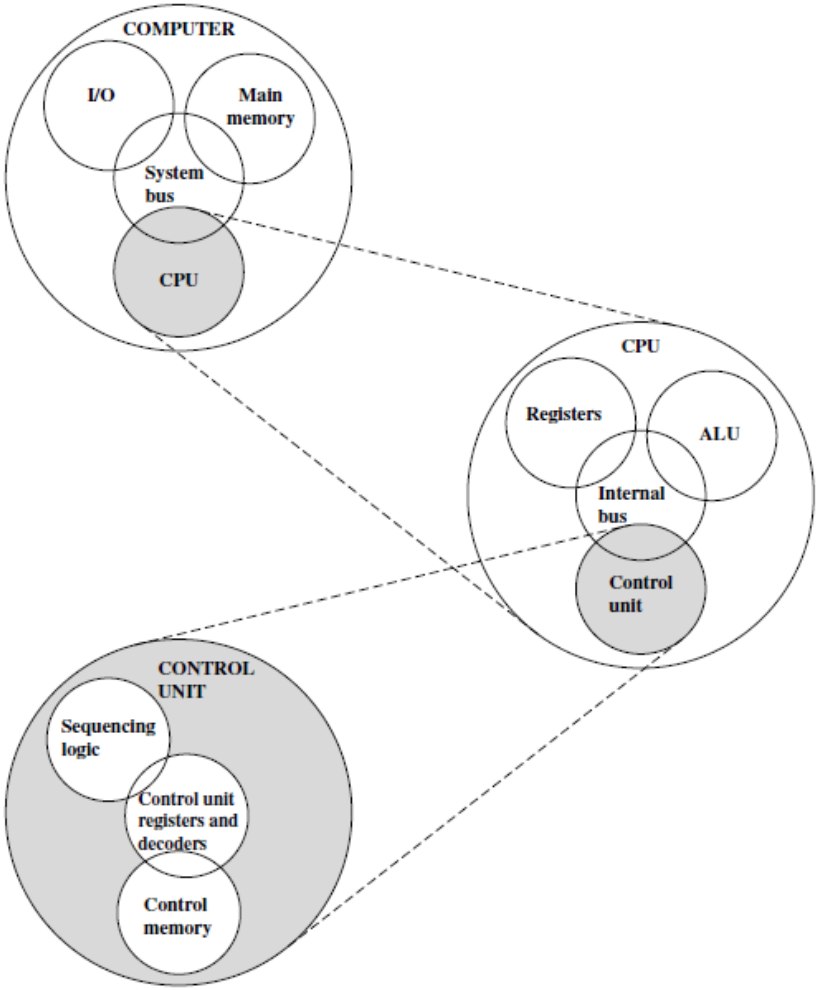
Basic functions that a computer performs:

- Data processing
- Data storage
- Data movement
- Control

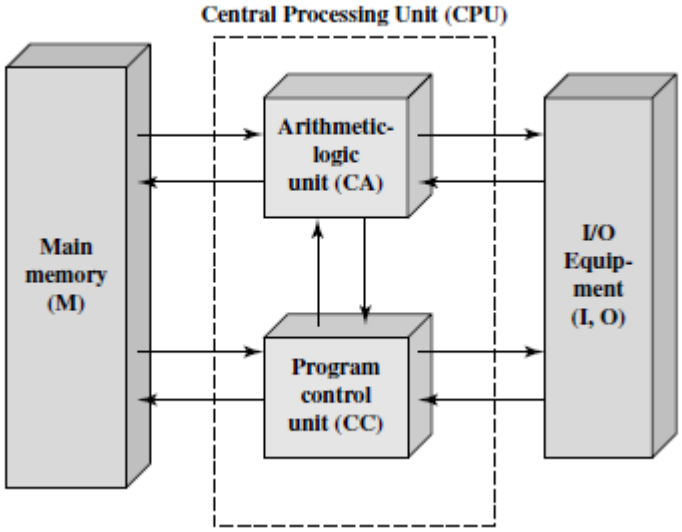


Structure of the computer





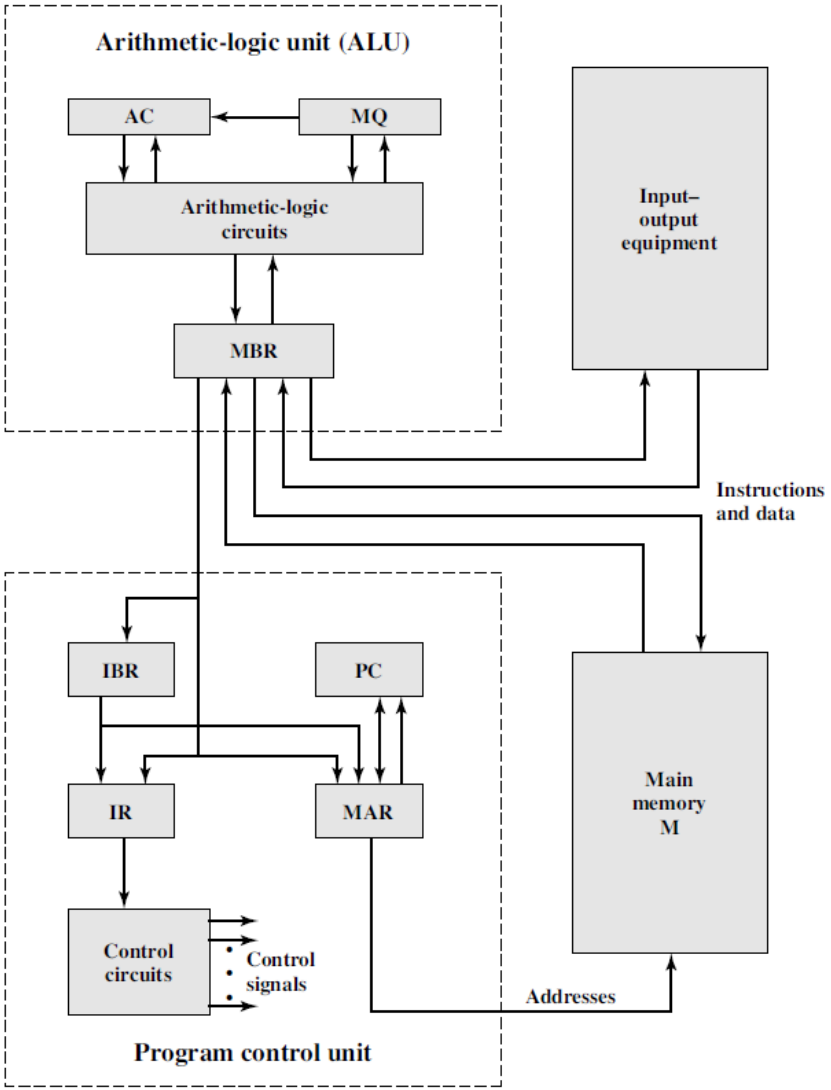
von Neumann machine



Computer consists of:

- A main memory, which stores both data and instructions
- An arithmetic and logic unit (ALU) capable of operating on binary data
- A control unit, which interprets the instructions in memory and causes them to be executed
- Input and output (I/O) equipment operated by the control unit

Computer components



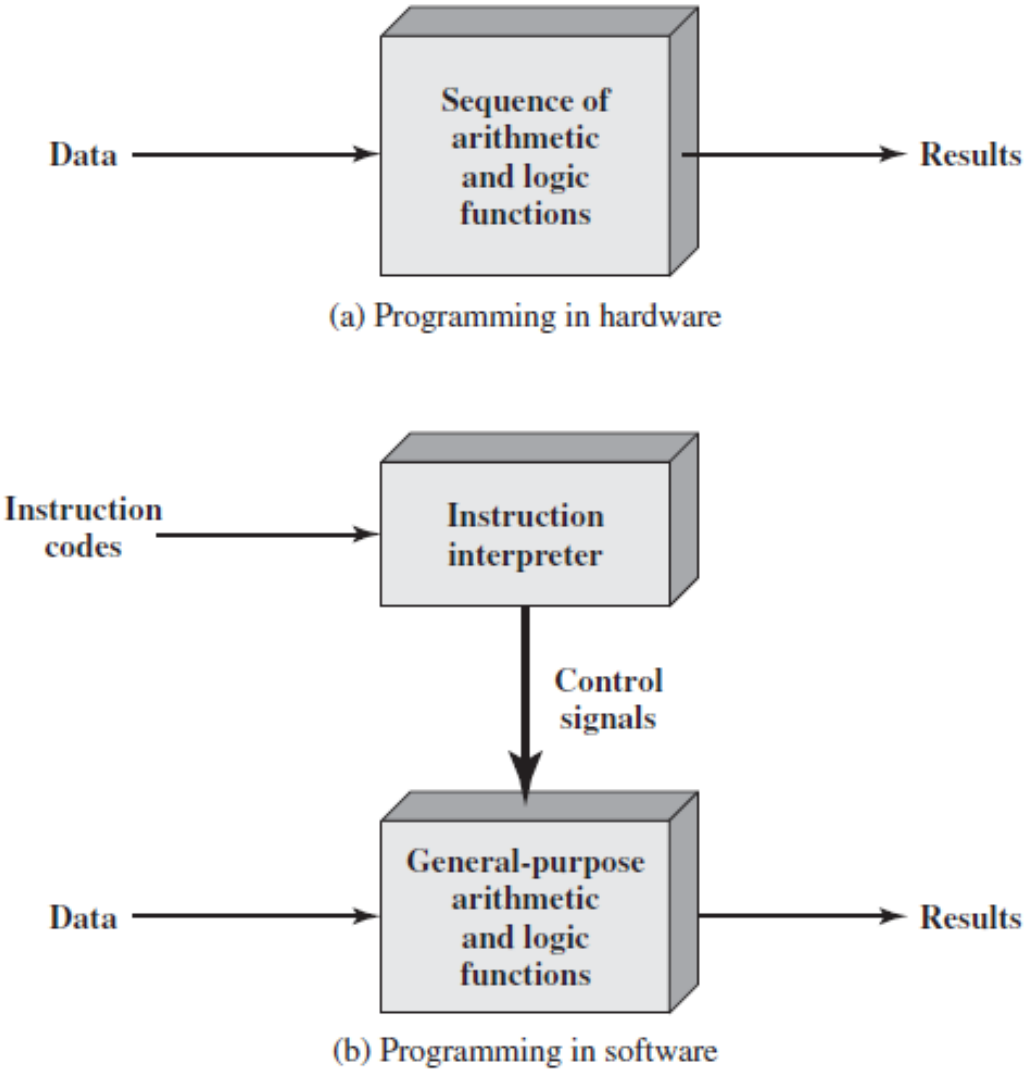
Registers

ALU contain storage locations, called *registers,* defined as follows:

- **Memory buffer register (MBR):** Contains a word to be stored in memory or sent to the I/O unit, or is used to receive a word from memory or from the I/O unit.
- **Memory address register (MAR):** Specifies the address in memory of the word to be written from or read into the MBR.
- **Instruction register (IR):** Contains the 8-bit opcode instruction being executed.
- **Instruction buffer register (IBR):** Employed to hold temporarily the right hand instruction from a word in memory.
- **Program counter (PC):** Contains the address of the next instruction-pair to be fetched from memory.
- **Accumulator (AC) and multiplier quotient (MQ):** Employed to hold temporarily operands and results of ALU operations.

Principle of the processor

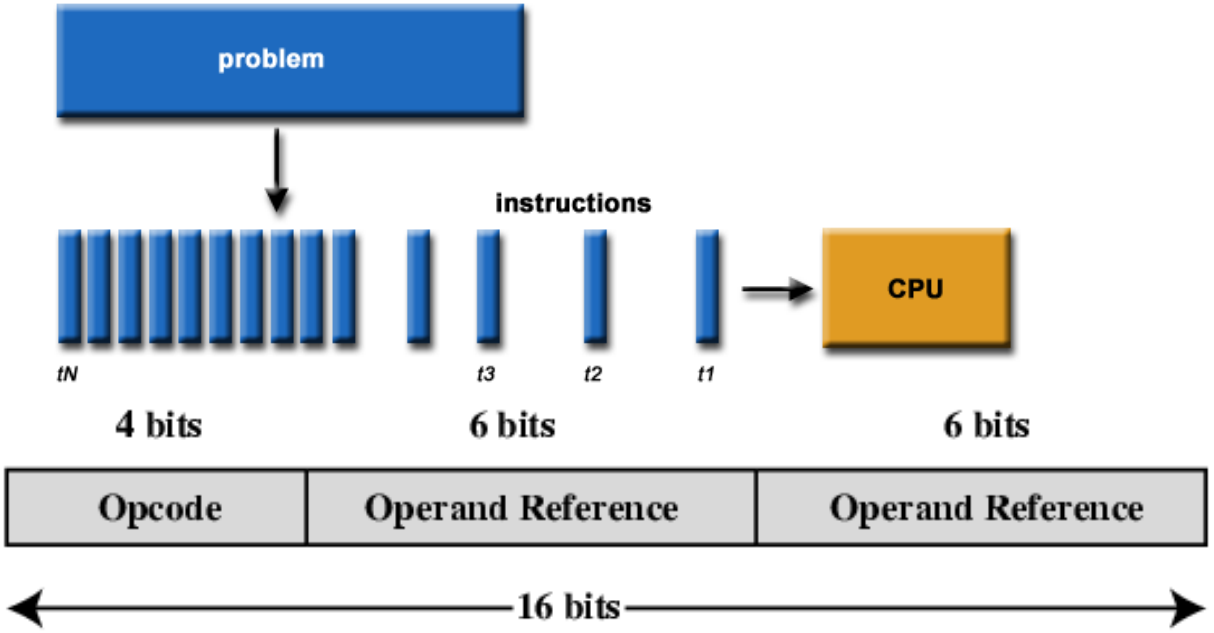
Hardware and Software Approaches



Operating principles

- **Execution of the program is the main function of the computer.**
- Program consists of serias of instructions and data that are located in main memory.
- CPU executes instructions.
- **Execution of the program is** continual cycles of the instruction fetch and execution.

Instruction



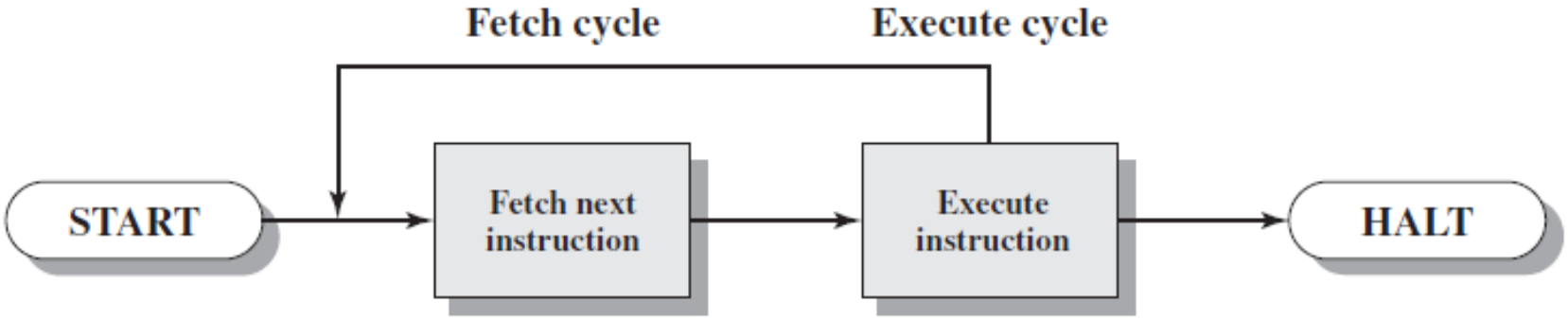
Instruction is an operation that is executed by processor.

Instructions

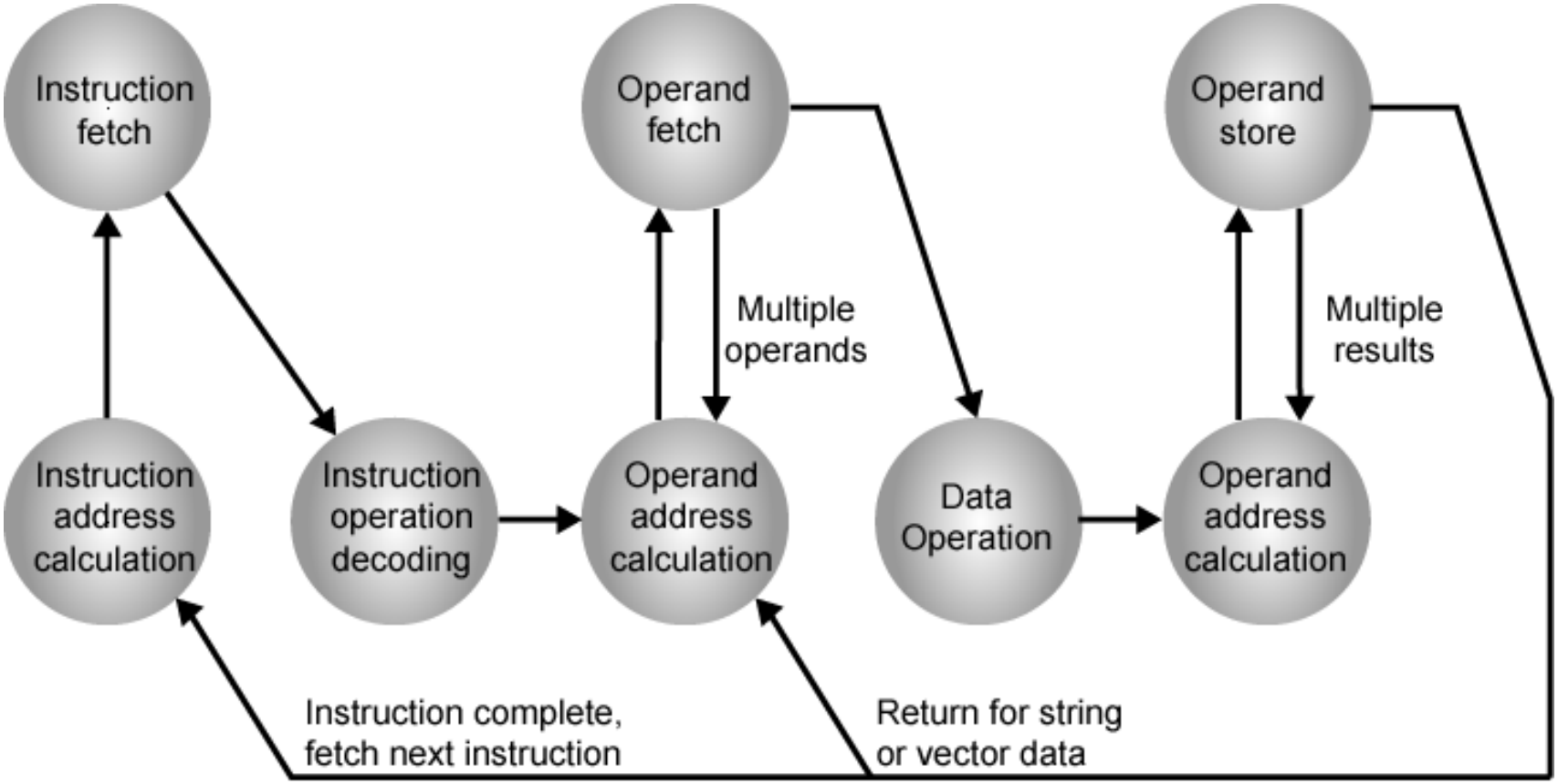
Instruction Type	Opcode	Symbolic Representation	Description
Data transfer	00001010	LOAD MQ	Transfer contents of register MQ to the accumulator AC
	00100001	STOR M(X)	Transfer contents of accumulator to memory location X
	00000001	LOAD M(X)	Transfer M(X) to the accumulator
	00000010	LOAD −M(X)	Transfer −M(X) to the accumulator
Unconditional branch	00001101	JUMP M(X,0:19)	Take next instruction from left half of M(X)
	00001110	JUMP M(X,20:39)	Take next instruction from right half of M(X)
Conditional branch	00001111	JUMP+ M(X,0:19)	If number in the accumulator is nonnegative, take next instruction from left half of M(X)
	00010000	JUMP+ M(X,20:39)	If number in the accumulator is nonnegative, take next instruction from right half of M(X)
Arithmetic	00000101	ADD M(X)	Add M(X) to AC; put the result in AC
	00000111	ADD |M(X)|	Add |M(X)| to AC; put the result in AC
	00000110	SUB M(X)	Subtract M(X) from AC; put the result in AC
	00001000	SUB |M(X)|	Subtract |M(X)| from AC; put the remainder in AC
	00001011	MUL M(X)	Multiply M(X) by MQ; put most significant bits of result in AC, put least significant bits in MQ
Address modify	00010010	STOR M(X,8:19)	Replace left address field at M(X) by 12 rightmost bits of AC
	00010011	STOR M(X,28:39)	Replace right address field at M(X) by 12 rightmost bits of AC

Instruction cycle

An **instruction cycle** consists of an instruction **fetch**, followed by zero or more operand fetches, followed by zero or more operand stores, followed by an interrupt check (if interrupts are enabled).

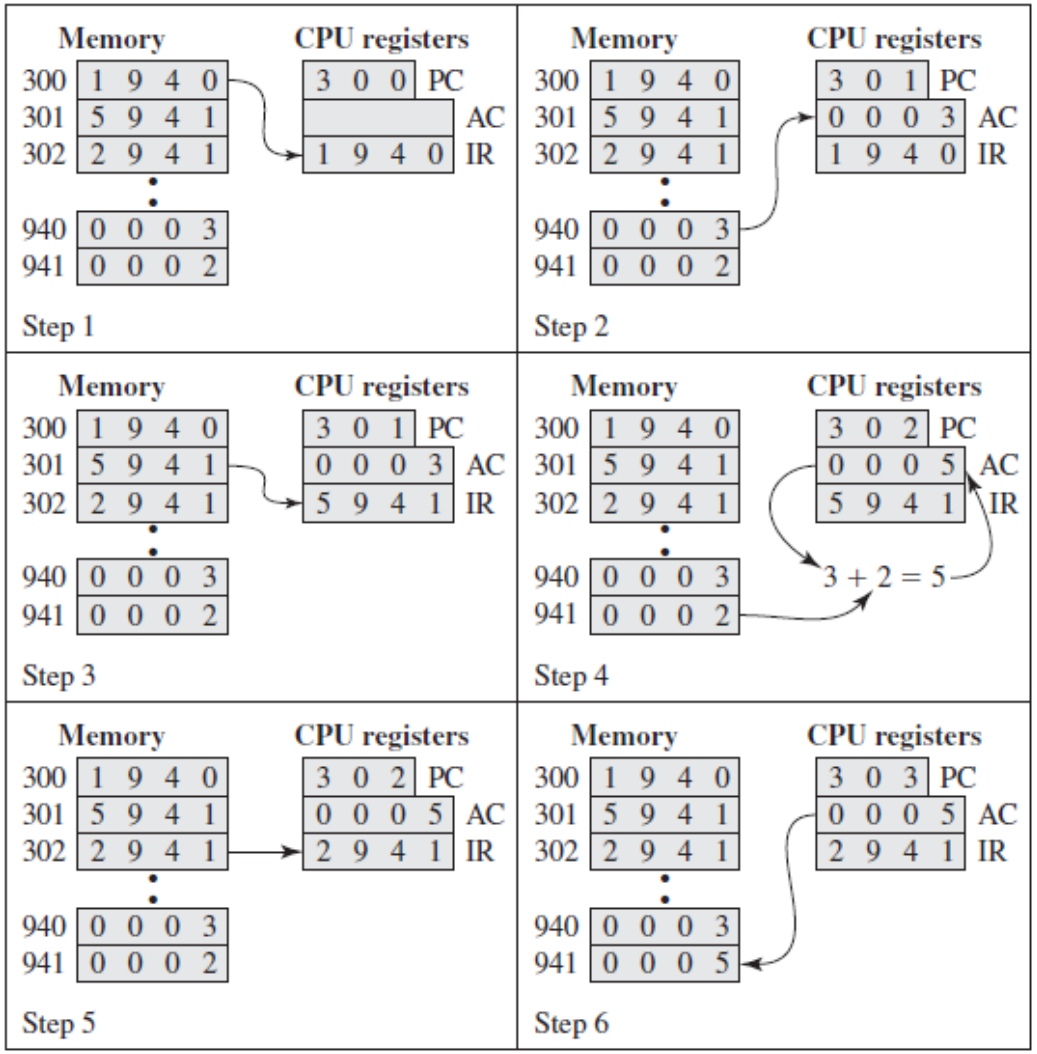


Instruction cycle state diagram



Example of program exection

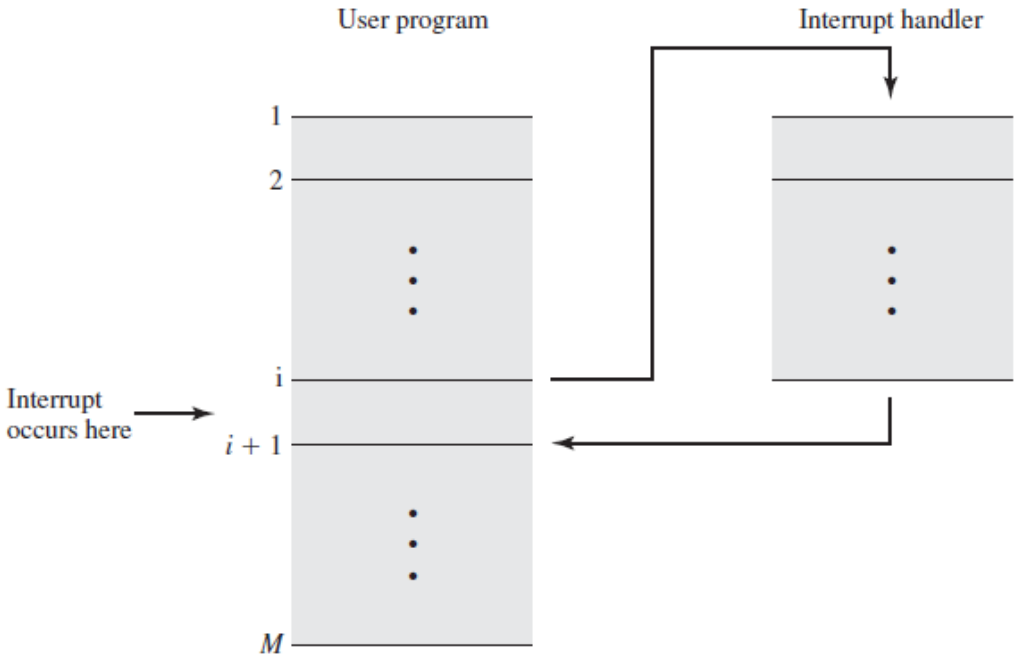
- Program counter (PC)
  - Address of instruction
- Instruction register (IR)
  - Instruction being executed
- Accumulator (AC)
  - Temporary storage



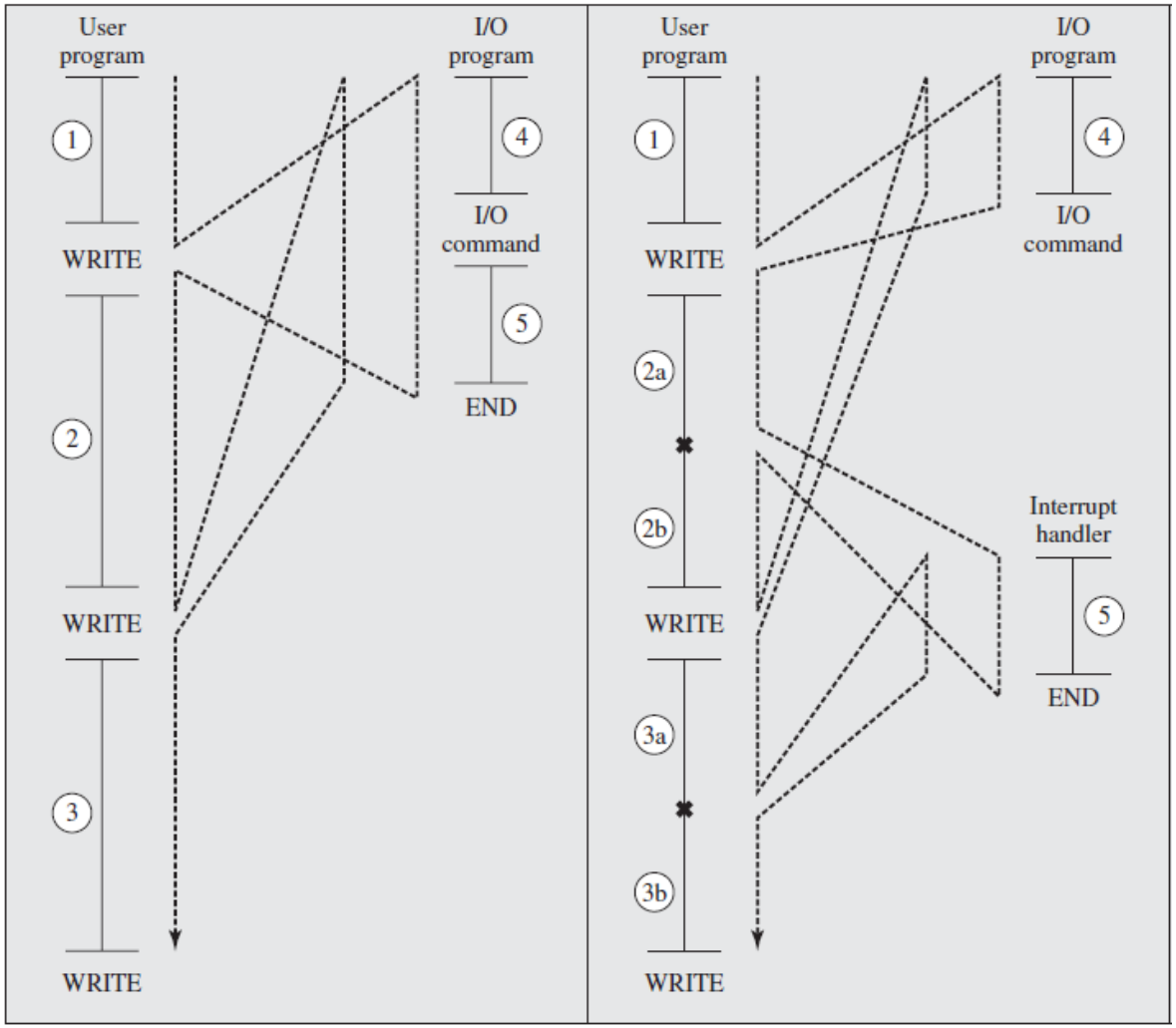
Interrupts

Virtually all computers provide a mechanism by which other modules (I/O, memory) may interrupt the normal processing of the processor.

Interrupts are provided primarily as a way to improve processing efficien



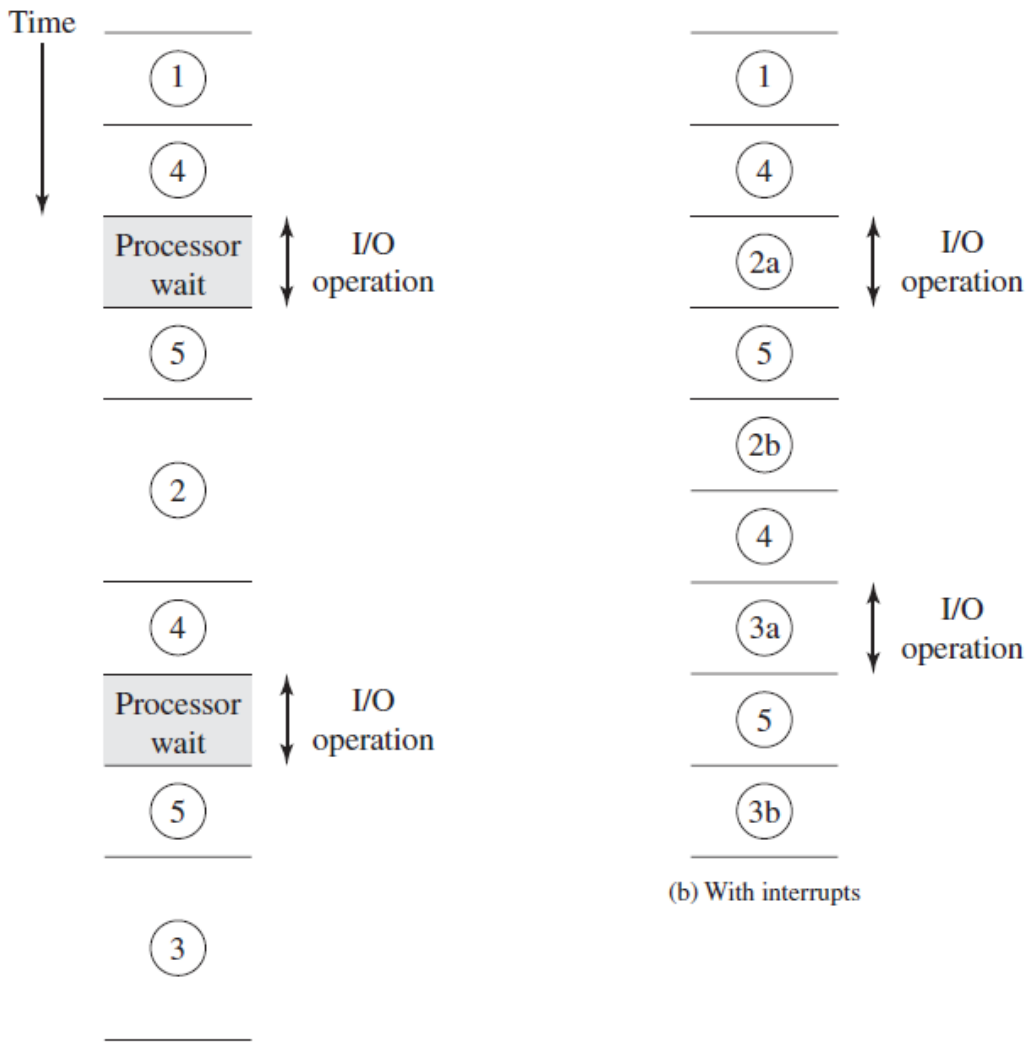
Program Flow



(a) No interrupts

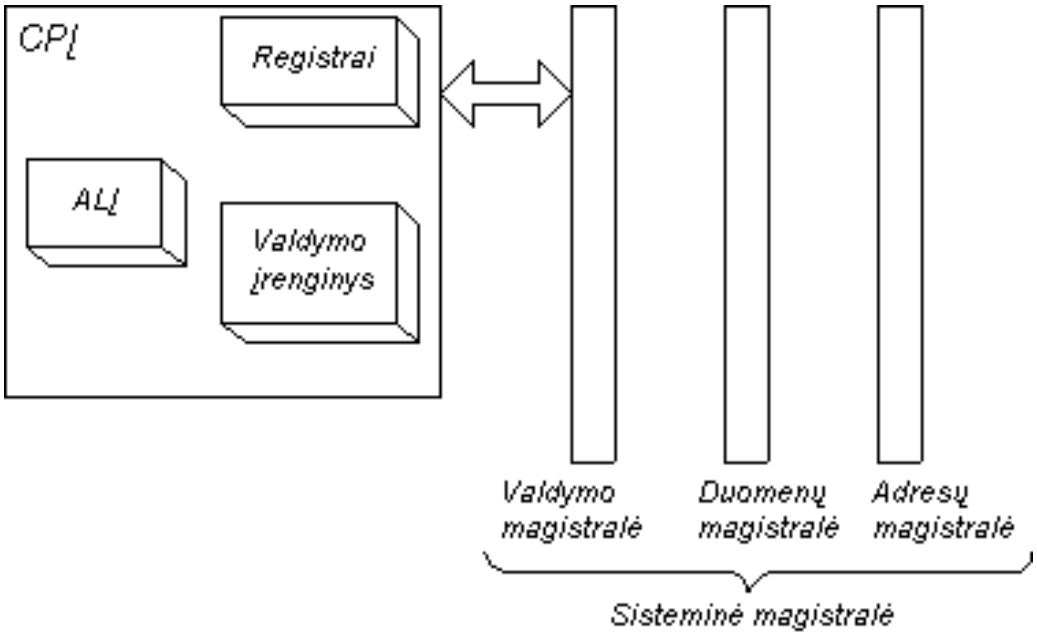
(b) Interrupts; short I/O wait

Program Timing

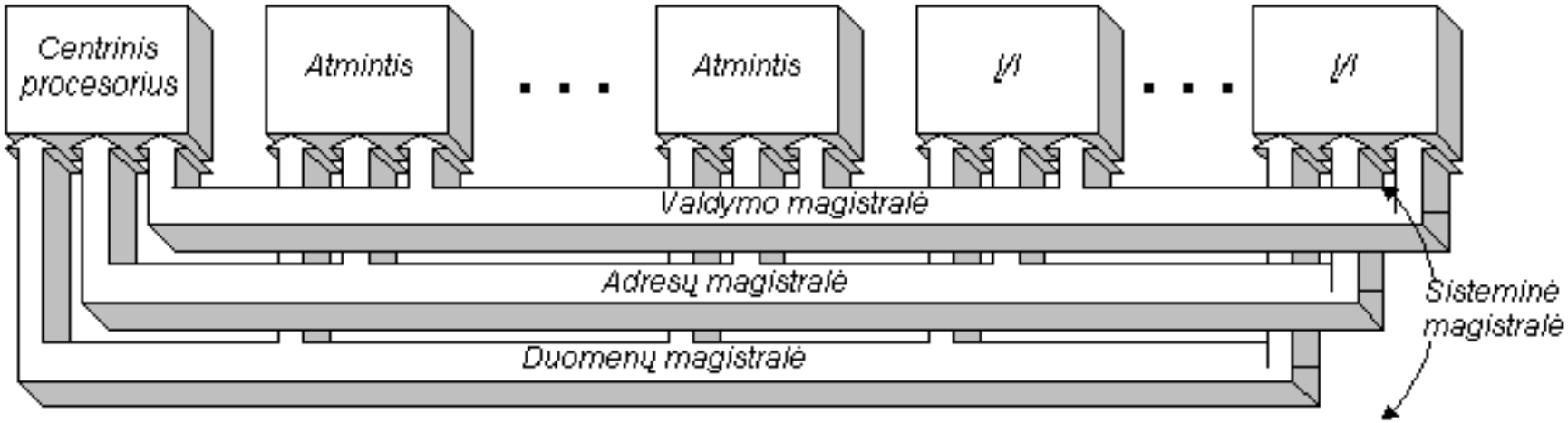


(b) With interrupts

Sisteminė magistralė



Magistralės struktūra



- Duomenų magistrale vyksta keitimasis duomenimis tarp kompiuterio modulių.
- Adresų magistrale nurodo duomenų magistralėje esančios informacijos šaltinį ir paskirties įrenginį (destination).
- Valdymo magistrale kontroliuoja kreiptis (access) į duomenų ir adresų linijas ir šių linijų naudojimą.

Registrų sandara

Registrų rinkinys funkcionuoja kaip atminties hierarchijos lygmuo, esantis aukščiau už pagrindinę ir spartinančiąją atmintis.

CPĮ registrai skirti dviem funkcijoms vykdyti:

• Vartotojo pasiekiami registrai. Šie registrai leidžia programuotojui, taikant asemblerio programavimo kalbą arba tiesiog per mašininius kodus, sumažinti kreipčių į pagrindinę atmintį skaičių ir optimizuoti šių registrų vartojimą.
• Valdymo ir būklės registrai. Jie naudojami CPĮ valdymo įrenginio darbui kontroliuoti, o taip pat valdyti operacinių sistemų programų taikomųjų programų vykdymą.

Vartotojo pasiekiami registrai

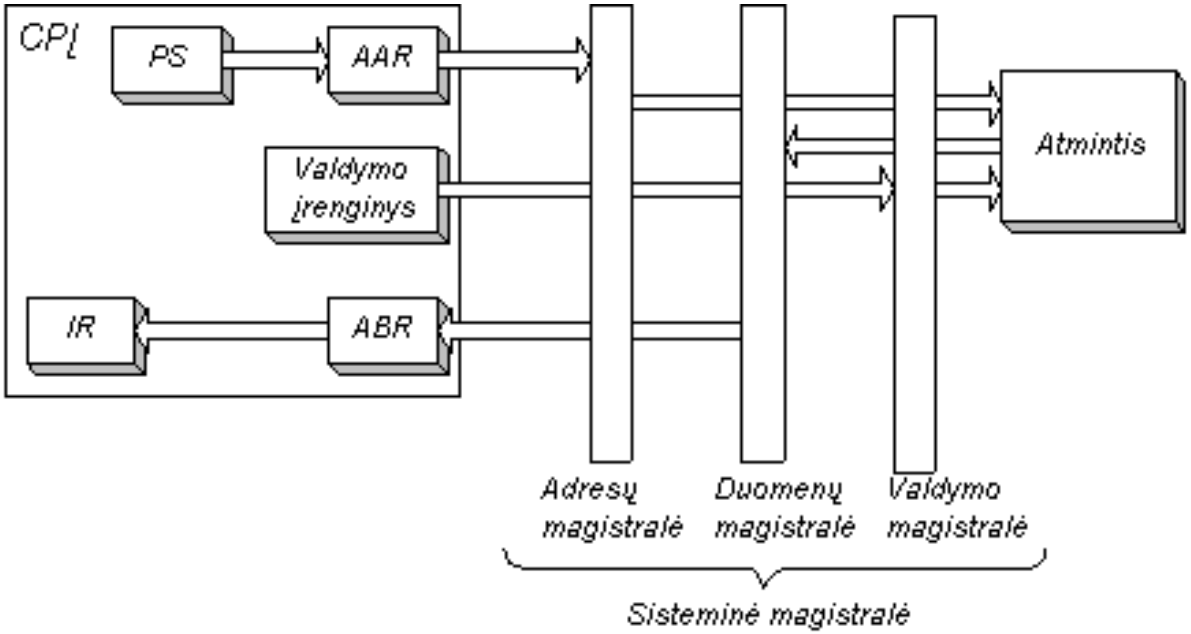
- • Bendrosios paskirties;
  • Duomenų;
  • Adresų
  • Sąlygų kodų.

- Bendrosios paskirties registrai programuotojo gali būti pritaikyti įvairioms funkcijoms.
- Duomenų registrai gali būti taikomi tik duomenims saugoti ir niekada netaikomi apskaičiuojant operandų adresus.
- Adresų registrai patys gali būti realizuojami iš bendrosios paskirties registrų, o gali būti skirti specialiems adresavimo metodams.
- Sąlygų kodus sudaro tam tikri bitai, kuriais gali operuoti CPĮ techninė įranga, priklausomai nuo vykdomos operacijos rezultato.

Valdymo būklės registrai

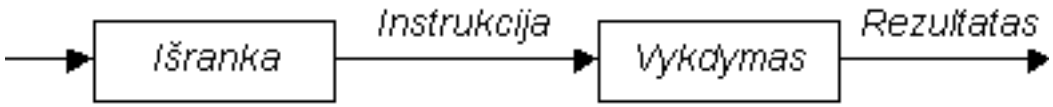
- Programos skaitiklis *(PS).* Saugo sekančios išrenkamos instrukcijos adresą.
- Instrukcijos registras *(IR).* Saugo naujausiąją išrinktą instrukciją.
- Atminties adreso registras *(AAR).* Saugo atminties ląstelės, iš kurios ką tik buvo perskaityta informacija arba į kurią vyks duomenų rašymas, adresą.
- Atminties buferio registras *(ABR).* Saugomas duomenų žodis, kuris bus įrašytas į atmintį arba iš atminties ką tik nuskaitytas.

Duomenų srauto diagrama

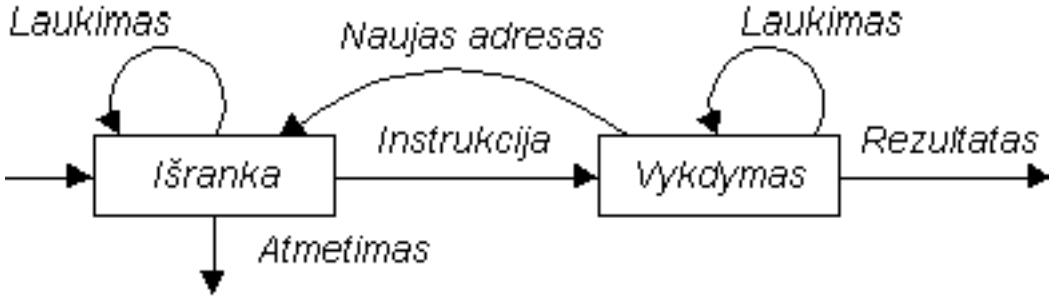


Instrukcija *ciklo išrankos* metu skaitoma iš atminties. PS registre saugomas sekančios išrenkamos instrukcijos adresas. Šis adresas siunčiamas į AAR ir patalpinamas adresų magistralėje. Valdymo įrenginys organizuoja skaitymą iš atminties. Nuskaitytas rezultatas patalpinamas duomenų magistralėje, kopijuojamas į ABR ir toliau siunčiamas į IR. Tuo tarpu PS turinys didinamas vienetu ir taip gaunamas sekančios instrukcijos adresas.

Instrukcijų konvejeris



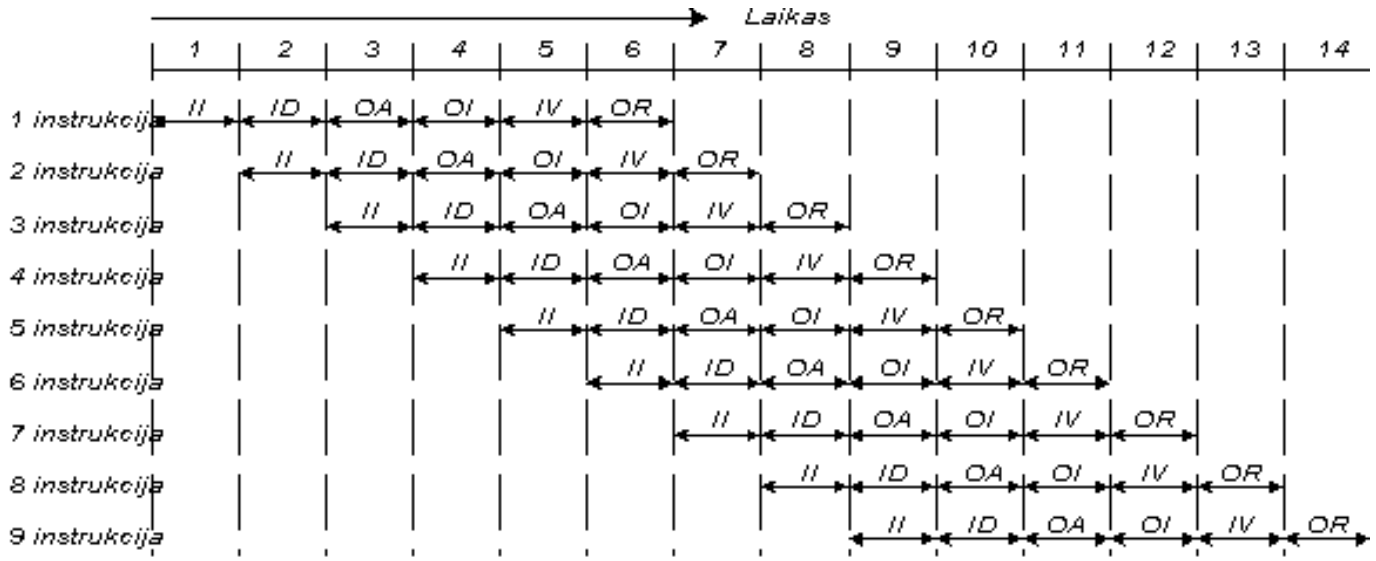
a) Supaprastinta schema



b) Išsamesnė schema

Instrukcijų konvejerizavimas panašus į surinkimo linijos taikymą produkcijos gamyboje. Surinkimo linijos privalumas yra tame, kad surinkimo procesas surinkimo linijoje išskaidomas į etapus ir tame, kad tam tikros gaminio surinkimo procedūros gali būti vykdomos vienu metu. Toks procesas dar vadinamas *konvejeriu* arba *vamzdynu* (pipeline), kadangi konvejeryje nauja operacija pradedama viename gale nesulaukus kol kita operacija kitame gale bus užbaigta.

Instrukcijų konvejeris



Instrukcijos išranka (II). Į buferį nuskaitoma sekanti instrukcija po vykdomos.
Instrukcijos dekodavimas (ID). Nustatomi operacijos kodas ir operando specifikatorius.
Operando apskaičiavimas (OA). Apskaičiuojamas operando efektyvusis (realusis arba fizinis) adresas. Tai gali būti perkėlimo, registrų netiesioginis adresavimo ir kitų formų adresų apskaičiavimai.
Operandų išranka (OI). Operandas arba operandai nuskaitomas iš atminties.
Instrukcijos vykdymas (IV). Vykdoma instrukcijoje nurodyta operacija ir išsaugomas jos rezultatas.
Operando rašymas (OR). Operacijos rezultatas įrašomas į atmintį.

Konvejerio našumas problemos

Konvejerio našumą įtakojančius faktorius priimta skirstyti į tris klases:

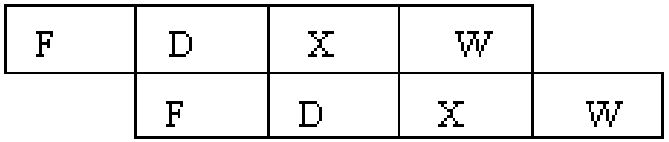
1) resursų konfliktus (*structural hazards*);
2) duomenų priklausomumą (*data hazards*);
3) valdymo nuoseklumo priklausomumą (*control hazards*).

Resursų konfliktai kyla tuomet, kai į tą patį vienintelį resursą kreipiasi ne mažiau kaip dvi komandos. Tokiu atveju viena komanda gali tuo resursu naudotis, o likusios komandos turi laukti, kol resursas bus atlaisvintas ir bus patenkinta kitos komandos paraiška. Jei to tipo resursas turi **k** egzempliorių (pavyzdžiui, superskaliariniame procesoriuje yra keli to paties tipo funkciniai įtaisai), aišku, kad vienu metu gali būti patenkinta **k** kreipinių.

Konvejerio našumas problemos

Duomenų priklausomumas atsiranda dėl to, kad konvejerizuotame procesoriuje pasikeičia gretimų komandų atskirų žingsnių vykdymo nuoseklumas. Tai liečia operandų išrinkimą ir rezultato įrašymą. Pavyzdžiui, vykdoma tokia komandų pora:

```
Add R1, R2, R3  ;  R1 ← R2 + R3
Sub R4, R4, R1  ;  R4 ← R4 + R1
```



Komandos **Add** rezultatas bus įrašytas į registrą R1 fazėje W , o komanda **Sub** atėminį turi išrinkti iš registro R1 fazėje F , kuri vykdoma dviems ciklais anksčiau.

Šis pavyzdys atitinka vadinamąjį **tikrąjį duomenų priklausomumą.**

Valdymo problemos konvejeryje

- II, OI ir OR etapuose vykdomi kreipiniai į atmintį. Diagramoje numatyta, kad visos šios kreiptys gali vykti vienu metu.
- Instrukcijos etapų trukmės skirtumai;
- Sąlygiškojo šakojimosi instrukcija;
- Tam tikrą neapibrėžtumą taip pat įneša pertrauktys.

Klausimai?