








# ЛЕКЦИЯ 4

## ЦЕНТРАЛЕН ПРОЦЕСОР

-  **Предназначение и състав**
-  **Организация на различни ЦП**
-  **Изпълнение на МИ**
-  **Видове УУ**
-  **Видове машинни езици**
-  **Паралелни архитектури**
-  **Кеш памет**

КА - 04

1/20


## ПРЕДНАЗНАЧЕНИЕ И СЪСТАВ НА ЦП

ЦП е **предназначен да разбере** зададената му **програма** и да я **изпълни**.

**Съвкупността от МИ**, които разпознава един ЦП, се нарича негов **машинен език (МЕ)**.

В **състава** на ЦП участват три устройства:

 **Управляващо Устройство – УУ (Control Unit – CU)**, знае МЕ и организира изпълнение на МП.

 **Аритметико-Логическо Устройство – АЛУ (Arithmetic and Logical Unit – ALU)**, умее да изчислява (събиране, конюнкция и др.).

 **Регистров блок** – временна памет в ЦП.

КА - 04

2/20

## РЕГИСТРИ НА ЦП

Регистрите на ЦП са два вида: **служебни** и **програмно достъпни** (за помнене на данни).

Служебни регистри на ЦП са поне следните:

- 🕒 **Програмен Брояч – ПБ (Program Counter – PC)**.
- 🕒 **Регистър на Инструкциите – РИ (Instruction Register – IR)**: съхранява изпълняваната МИ.
- 🕒 **Регистър за Адрес от Паметта – РАП (Memory Address Register – MAR)**: адрес на данни в ОП.
- 🕒 **Регистър на Условиата – РУ (Condition Code Register – CCR)**: регистрира дейността на АЛУ.

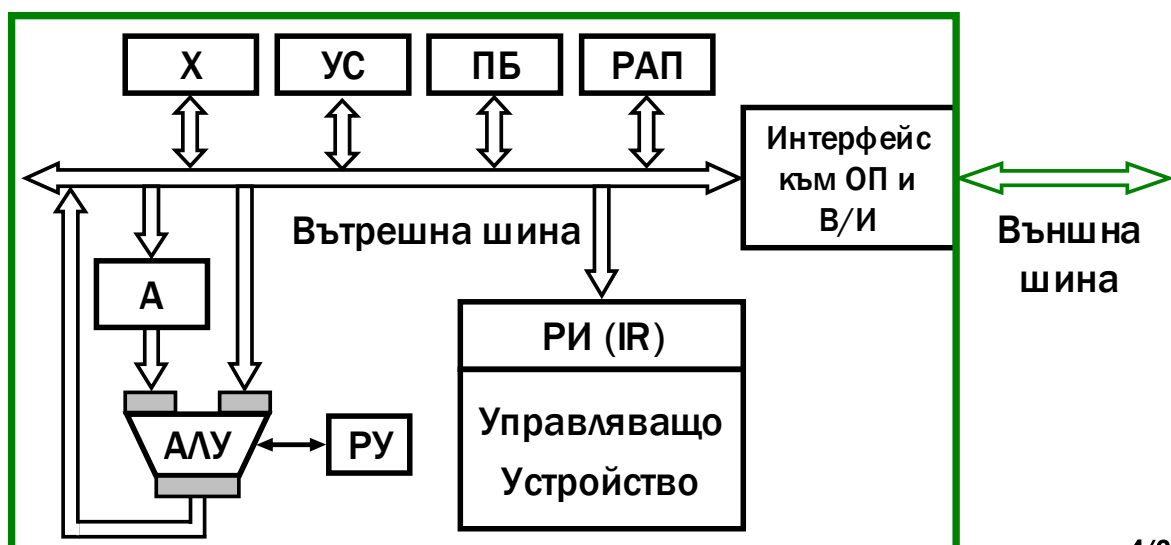
Главни регистри за данни са **акумулатор (А)**, **индексен (Х)** и **указател на стек (УС)**.

КА - 04

3/20

## АКУМУЛАТОРЕН ЦП

Най-проста организация има **акумулаторният** процесор. Той има **един или** (най-много) **два** **акумулатора** и **уникални други** регистри.



4/20



## СРАВНЯВАНЕ НА ЦП

### Акумулаторен:

- ☺ уникални регистри.
- ☹ за кое по-напред.
- ☹ много преноси между А и ОП.
- ☹ бавно изпълнение.
- ☺ не се посочват.
- ☺ къси инструкции.
- ☺ не се мисли кога, кой и как ще се използва.

### С РОП:

- ☺ универсални р-ри.
- ☺ достатъчен брой.
- ☺ има за всичко.
- ☺ ускорено изпълнение.
- ☹ трябва да се посочват чрез номерата си.
- ☹ трябва да се мисли за разпределянето им.
- ☹ не чак толкова много.

КА - 04

7/20

## ЦИКЪЛ НА УУ

За да се изпълни поредната машинна инструкция УУ извършва следното:

### ① Чете МИ от ОП (фаза извличане):

- 📖 адресна шина := ПБ (през интерфейса);
- 📖 сигнал четене/запис := четене;
- 📖 ПБ := ПБ + 1 (докато се чака отговор от ОП);
- 📖 РИ := даннова шина (през интерфейса).

### ② Дешифрира МИ в РИ (фаза декодиране).

### ③ МИ се изпълнява (фаза изпълнение):

- 📖 посредством РАП от ОП се четат операндите;
- 📖 АЛУ извършва необходимите изчисления;
- 📖 посредством РАП в ОП се записва резултата.

КА - 04

8/20

## ВИДОВЕ УУ

Описаната процедура се нарича **цикъл извлечи–декодирай–изпълни** на ЦП.

**Дума**, адресирана **чрез ПБ** във фаза **извличане**, се приема за записана **МИ**.

**Дума**, адресирана **чрез РАП** във фаза **изпълнение**, се приема за **данни**.

Фаза **изпълнение** на **управляваща МИ** просто **записва в ПБ нова стойност**.

Реализацията на **фаза изпълнение** дава:

**Апаратен ЦП** – включва се избрана **ел. схема**;

**Микропрограмируем ЦП** – следва се избрана **микропрограма**.

КА -04

9/20

## МАШИННИ ЕЗИЦИ

**ЦП** се проектират **от различни хора** ⇒  
техните **МЕ** се различават:

- ① по наличните **операции** (да **има** или да **няма** операция **умножение?**);
- ② по **кода** на една и съща **операция** (кое ще бъде код на операция **събиране: 15** или **62?**)

**В резултат** от това различие **МП** на един **ЦП** е абсолютно **неразбираема** за другите **ЦП**.

**Следствие** от този факт е **спирането на прогреса**: **сменим ли ЦП губим всички създадени за него МП**, т. е. **съвсем всичко!**

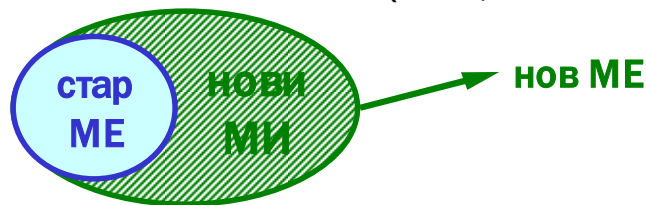
КА -04

10/20

## РЕШЕНИЕ НА ПРОБЛЕМА

За да се осигури възможност за **използване на вече създадените МП** има **2 пътя**:

- 1 **апаратна съвместимост** от долу на горе: нов МЕ се създава като **съществуващ МЕ** се **допълва с нови МИ** (IBM, Intel и др.);



- 2 **програмна емуляция**: **цикълът на УУ** извлечи-декодирай-изпълни има чисто алгоритмичен характер и може да **се моделира** и **по програмен път** на новия ЦП (**Power MAC**).

КА - 04

11/20

## ВИДОВЕ ЕЗИЦИ

В зависимост от МЕ се различават:

- 1 **Компютри с редуцирана система МИ** – **Reduced Instructions Set Computer (RISC)**: **всички МИ** са с **еднаква структура**, малък брой КОП, изпълнение на МИ за **1** цикъл.
- 2 **Компютри със сложна система МИ** – **Complex Instructions Set Computer (CISC)**: **МИ** са с **различна структура**, богат набор от сложни (и може би по-удобни) МИ.

**RISC** се реализират по **апаратен път**.

**CISC** се реализират по **микропрограмен път**.

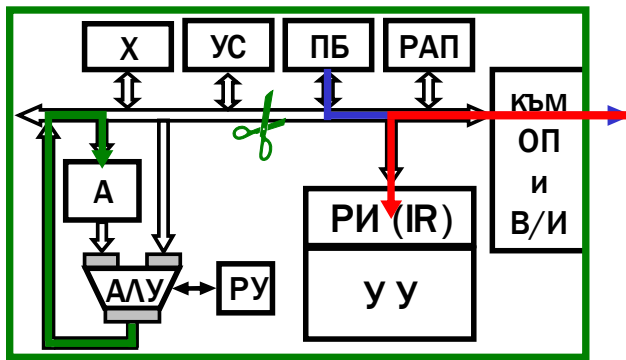
КА - 04

12/20

# ПАРАЛЕЛИЗЪМ

Производителността на ЦП може да бъде увеличена **над поставените** от природата **границы** само чрез **паралелна работа**.

Ето **най-простия** вид паралелизъм:



- ① **Край** (запис на получения резултат в акумулатора)
  - ② **Начало** (четене на новата МИ)
- Краят на една МИ и началото на следващата.**

КА - 04

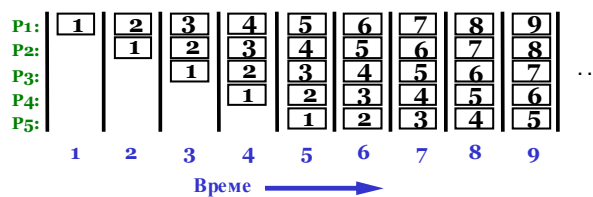
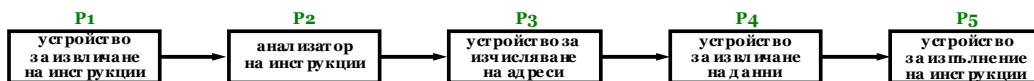
13/20

# КОНВЕЙЕРЕН ПРОЦЕСОР

Изпълнението на МИ има три фази, от които **последната е поне от три стъпки – общо 5**.

Нека **последователен** процесор **изпълнява всяка една от тях за 1 такт**.

Ако изработим **УУ от 5 секции, които си сътрудничат** при изпълнение, ще получим:



КА - 04

14/20

## ПАРАЛЕЛНИ МАШИНИ

**Флин (Flynn, 1982)** разделя паралелните машини на **три категории** на база на това, **колко потока от инструкции и данни имат:**

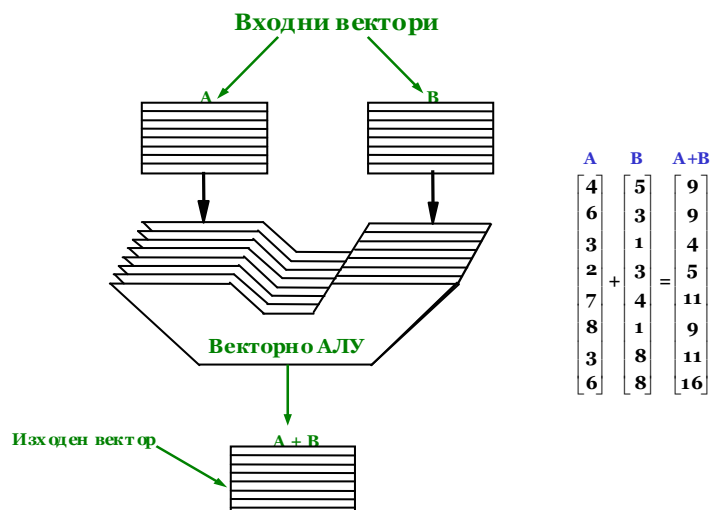
- ① **ЕИЕД:** Един поток Инструкции, Един поток данни (**SISD**, Single Instruction stream, Single Data stream).
- ② **ЕИМД:** Един поток Инструкции, Много потоци Данни (**SIMD**, Single Instruction stream, Multiple Data stream).
- ③ **МИМД:** Много потоци Инструкции, Много потоци Данни (**MIMD**, Multiple Instruction stream, Multiple Data stream).

КА - 04

15/20

## ВЕКТОРЕН ПРОЦЕСОР

Представители на **ЕИМД** са **векторните** и **матричните** процесори. Те оперират едновременно над множество данни.



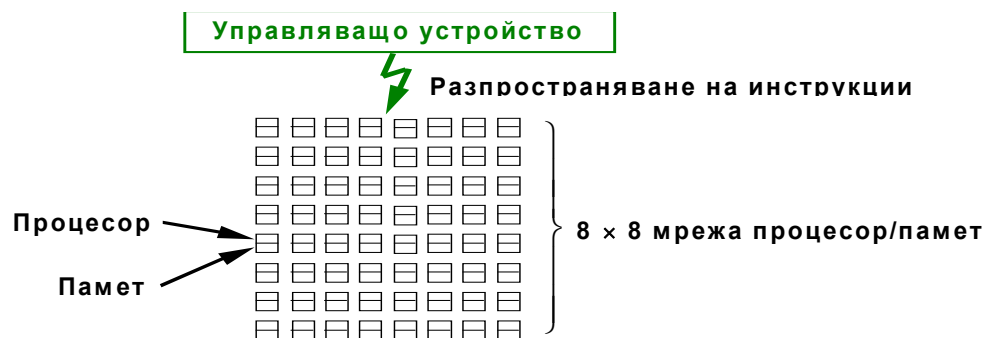
КА - 04

16/20



## МАТРИЧЕН ПРОЦЕСОР

За първи път **матричен процесор (array processor)** е проектиран в компютъра **ILLIAC IV** на Илинойския Университет. Тази архитектура се състои от **квадратна мрежа от елементи процесор/памет.**



КА - 04

17/20

## КЕШ ПАМЕТ

**Паралелизмът в ЦП повишава производителността, но решаващ фактор за нея си остава бързодействието на ОП.**

Уви, колкото **по-бърза** е **ОП**, толкова **по-висока** е и нейната **цена.**

В 60-те години биват открити алгоритми, които позволяват **комбинация от голяма по обем, но бавна памет, и малка по обем, но бърза памет, да работят почти със скоростта на бързата памет.**

КА - 04

18/20

## КЕШ ПАМЕТ (прод.)

Днес **модификация на тези алгоритми** се реализира **по апаратен път** от ЦП.

**Бързата памет** играе роля на **буфер между ЦП и ОП** и се нарича **кеш-памет (cache)**.

Изгодна стратегия е **кеш-паметта** да бъде **асоциативна**.

Много микропроцесори използват дори **два комплекта кеш**: за данни и за инструкции.

**Кеш-паметта** е два вида: **вътрешна** (в кристала на ЦП) и **външна**.

КА-04

19/20

**БЛАГОДАРЯ ВИ  
ЗА ВНИМАНИЕТО!**

**БЪДЕТЕ С МЕН И В  
СЛЕДВАЩАТА ЛЕКЦИЯ,  
КОЯТО ЩЕ НИ ОТВЕДЕ  
В НЕВЕРОЯТНИЯ СВЯТ НА  
РАЗЛИЧНИТЕ ВИДОВЕ  
АДРЕСИРАНЕ**