АРХИТЕКТУРА НА СИСТЕМАТА ИНСТРУКЦИИ

ПРОФ. Д-Р ПЛАМЕНКА БОРОВСКА

ПРЕДСТАВЯНЕ НА ДАННИТЕ
БРОЙНИ СИСТЕМИ

ПРИ ПОЗИЦИОННИТЕ БРОЙНИ СИСТЕМИ ЧИСЛАТА СЕ ПРЕДСТАВЯТ КАТО ПОЛИНОМ ОТ ВИДА:

$$A = \sum_i A_i * S_i \qquad i \in [-m, +n]$$

КЪДЕТО A_i Е ЦИФРАТА В i -ТАТА ПОЗИЦИЯ НА ЧИСЛОТО, S - ОСНОВАТА НА БРОЙНАТА СИСТЕМА, А ИНДЕКСЪТ i ПОКАЗВА ПОЗИЦИЯТА НА ЦИФРАТА В ЧИСЛОТО.

$$A = A_n S^n + A_{n-1} S_{n-1} + A_{n-2} S_{n-2} + \ldots + A_1 S^1 + A_0 S^0 +$$

$$+ A_{-1} S^{-1} + A_2 S^{-2} + \ldots A_{-m} S^{-m}$$

ОСНОВАТА ВИНАГИ Е $S=10=1.S^1+0.S^0$

НАПР. ПРИ S=10

$55555 = 5.10^4+5.10^3+5.10^2+5.10^1+5.10^0$

НАПР. ПРИ S=2

$1110_{(2)} = 1.2^3+1.2^2+1.2^1+1.2^0 = 14_{(10)}$

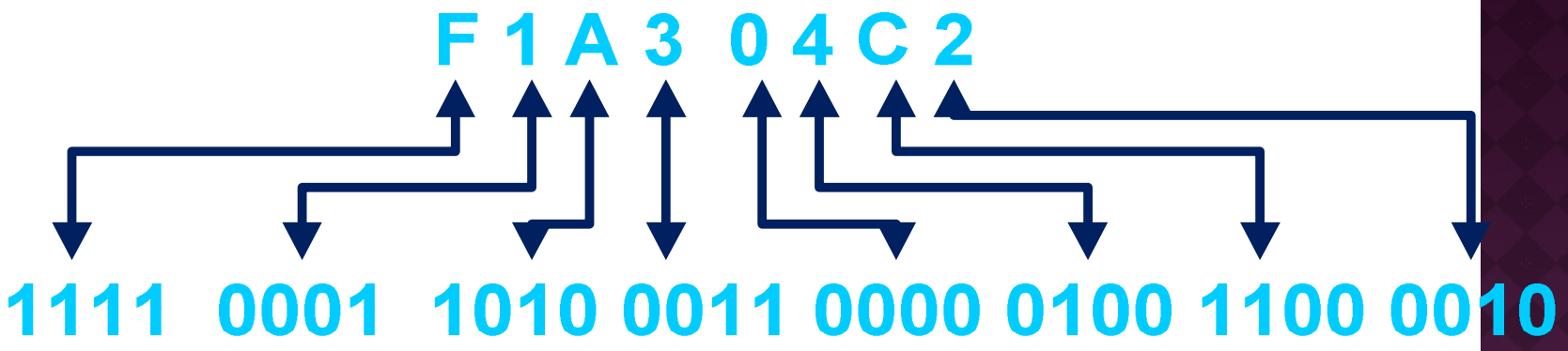
НАПР. ПРИ S=16

$1F_{(16)} = 1.16^1+15.16^0 = 31_{(10)}$

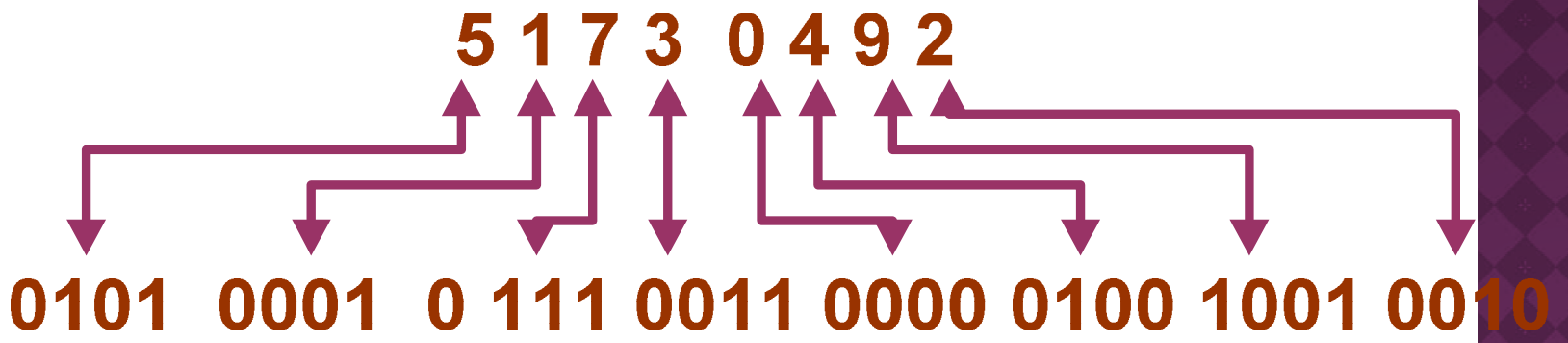
НАПР. ПРИ S=8

$123_{(8)} = 1.8^2+2.8^1+3.8^0=83_{(10)}$

ПРЕОБРАЗУВАНЕ НА ЧИСЛАТА ОТ ШЕСТНАЙСЕТИЧНА В ДВОИЧНА БРОЙНА СИСТЕМА И ОБРАТНО



Преобразуване на число от двоична в десетична бройна система и обратно при пакетирано десетично кодиране



ФОРМАТ НА ДАННИТЕ С ФИКСИРАНА ТОЧКА

ФОРМАТ НА ДАННИТЕ С ПЛАВАЩА ТОЧКА

$$A = \pm M * B \pm E$$

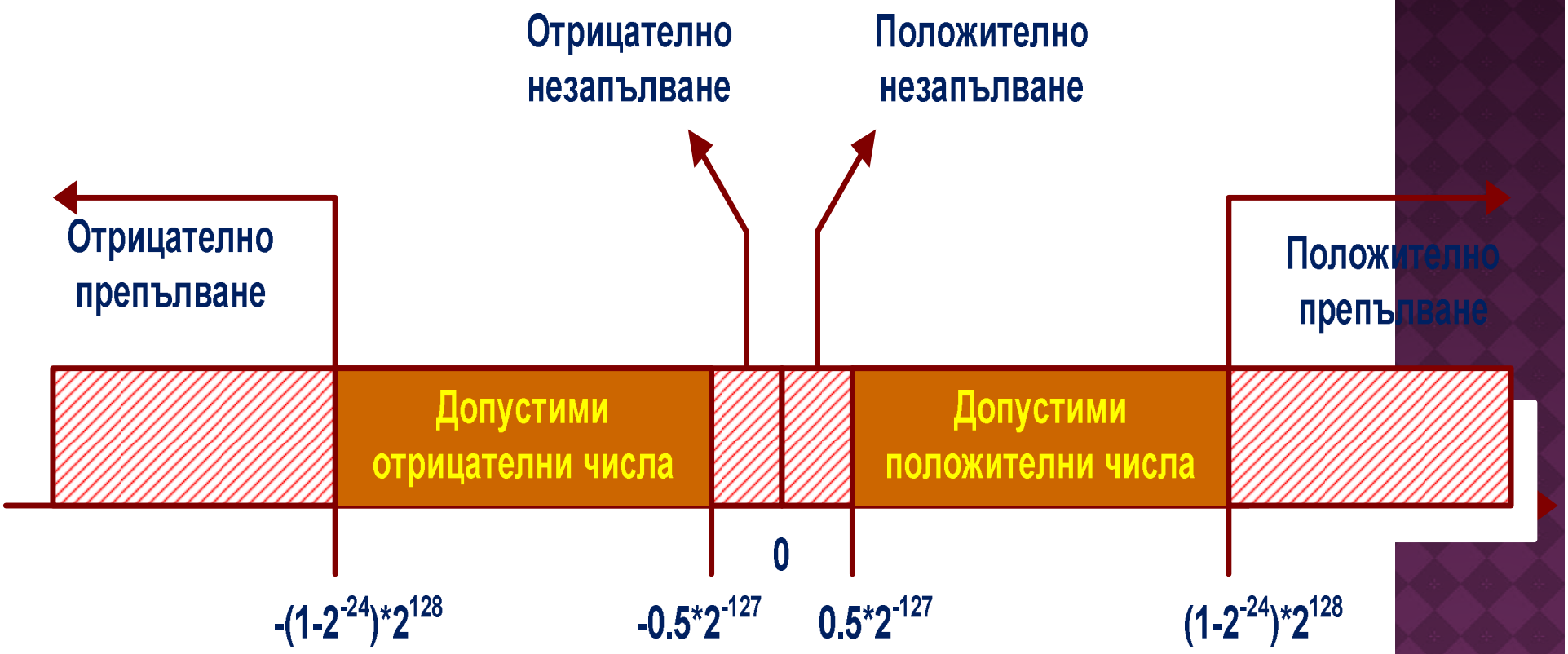
M - МАНТИСА

E - ПОРЯДЪК

B - ОСНОВА

(= ЗА ВСИЧКИ ЧИСЛА, НЕ СЕ ЗАДАВА В ЯВЕН ВИД)

ОБХВАТ НА ПРЕДСТАВЯНИТЕ ЧИСЛА С 32-БИТОВ ФОРМАТ С ПЛАВАЩА ТОЧКА IEEE STANDARD 754



ПРЕДСТАВЯНЕ НА СИМВОЛИ

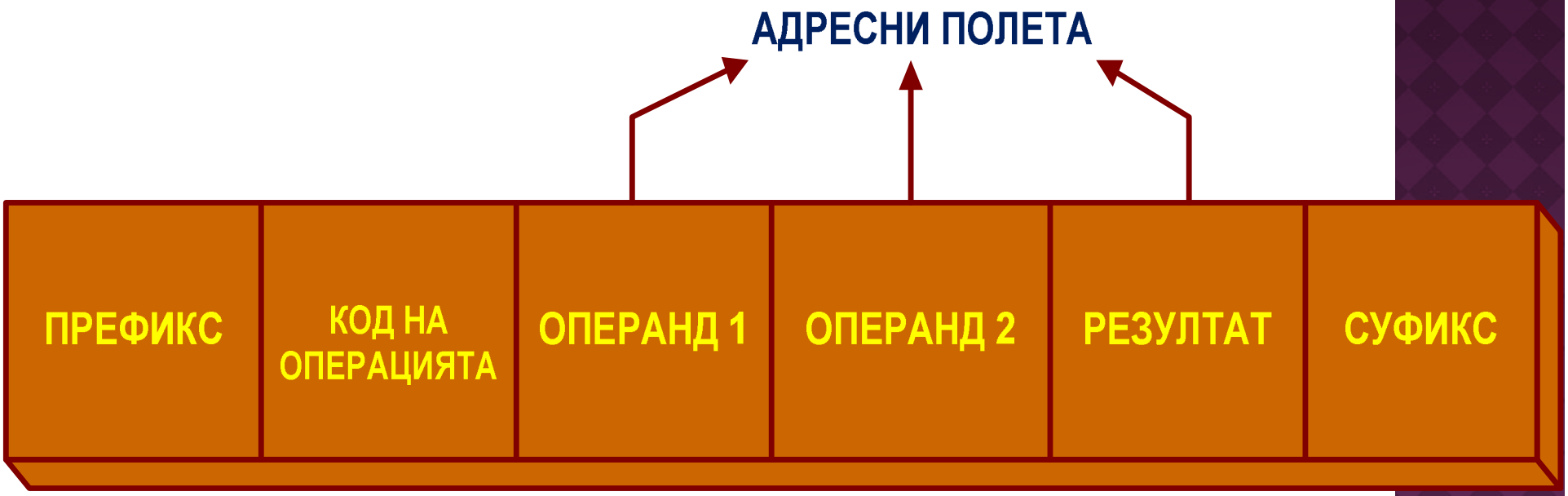
АМЕРИКАНСКИ СТАНДАРТЕН КОД ЗА ИНФОРМАЦИОНЕН ОБМЕН ASCII (AMERICAN STANDARD CODE FOR INFORMATION INTERCHANGE)

EBCDIC (EXTENDED BINARY CODED DECIMAL INTERCHANGE CODE)

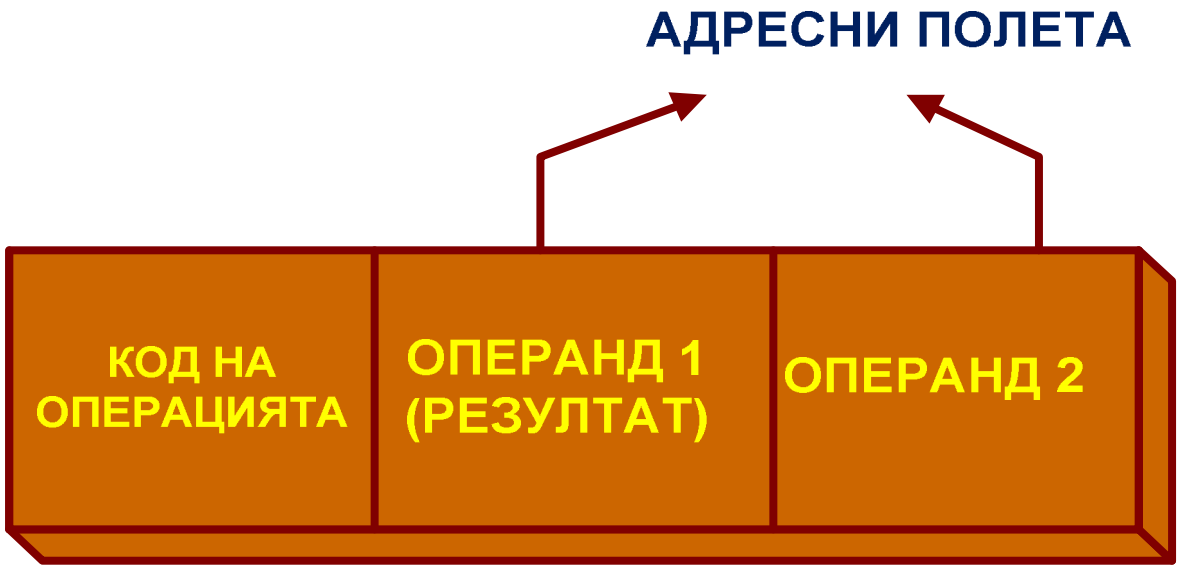
UNICODE

АРХИТЕКТУРА НА СИСТЕМАТА ИНСТРУКЦИИ

ФОРМАТ НА ИНСТРУКЦИИТЕ



Триадресен формат на компютърна инструкция

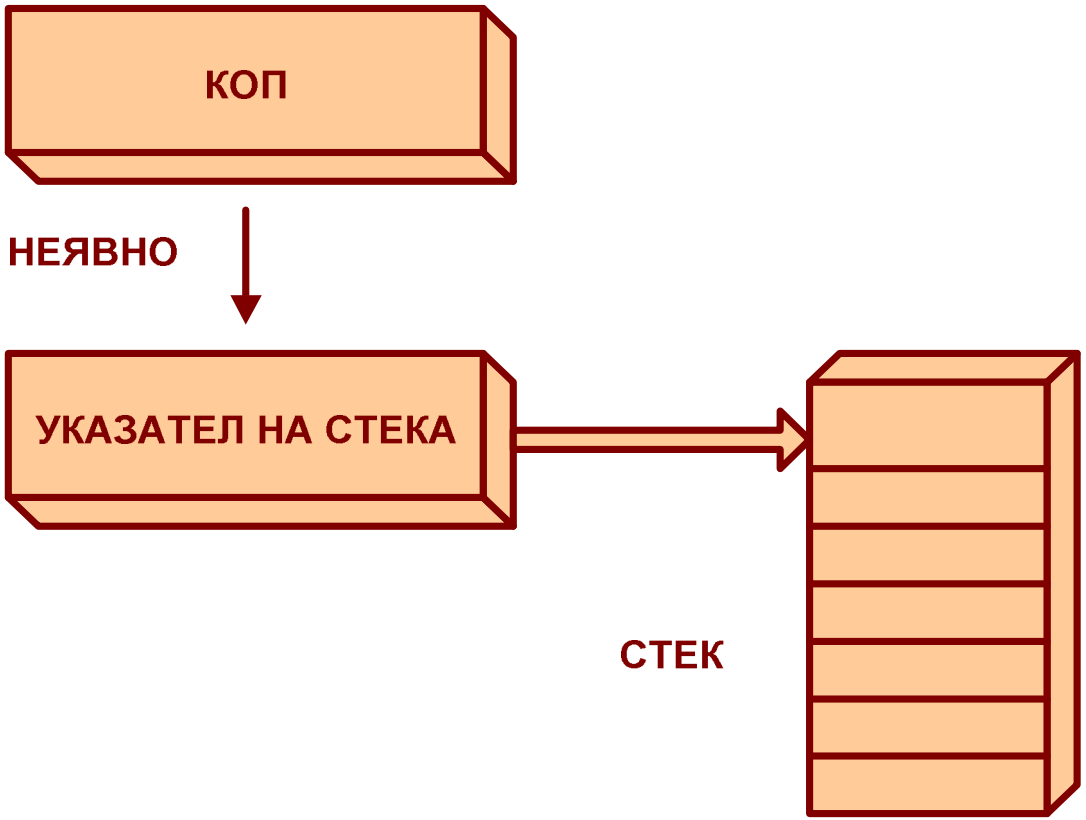


Двуадресен формат на компютърната инструкция

Едноадресен формат на инструкциите

РЕЖИМИ НА АДРЕСИРАНЕ

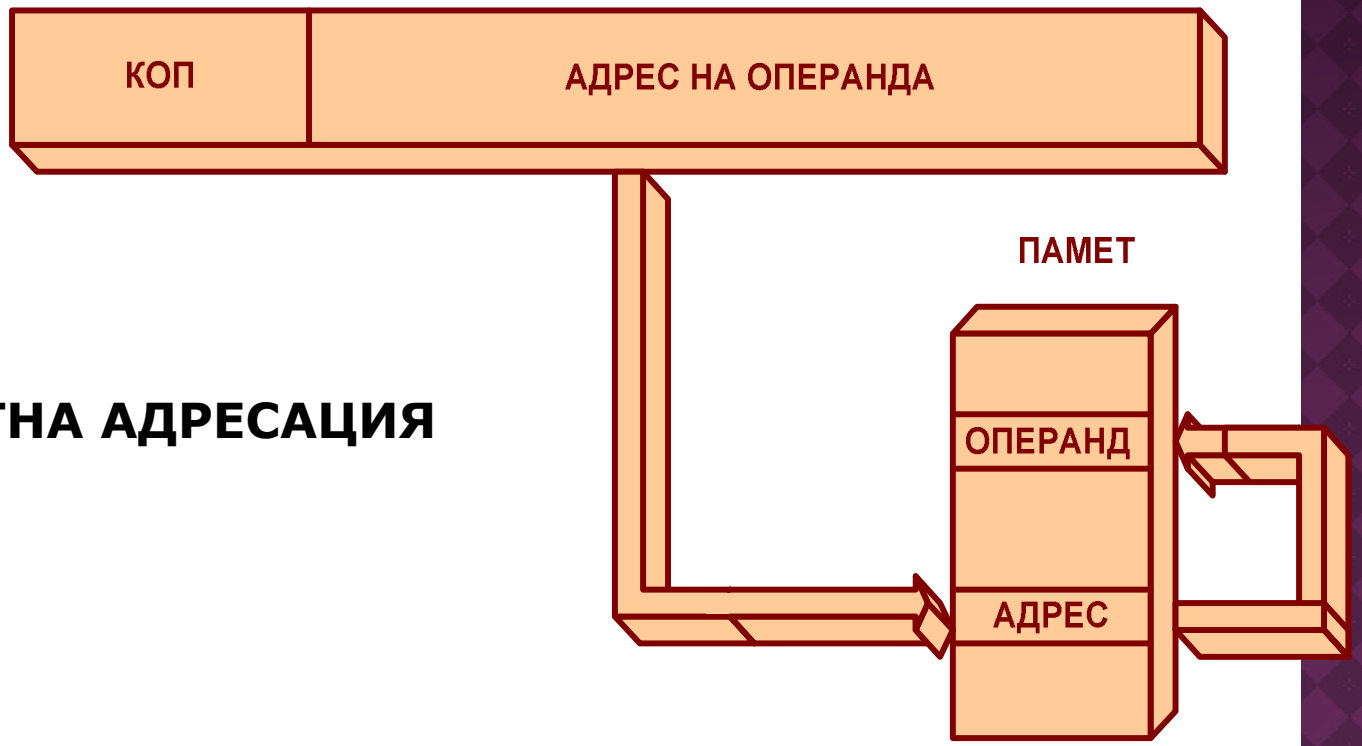
НЕПОСРЕДСТВЕНО АДРЕСИРАНЕ

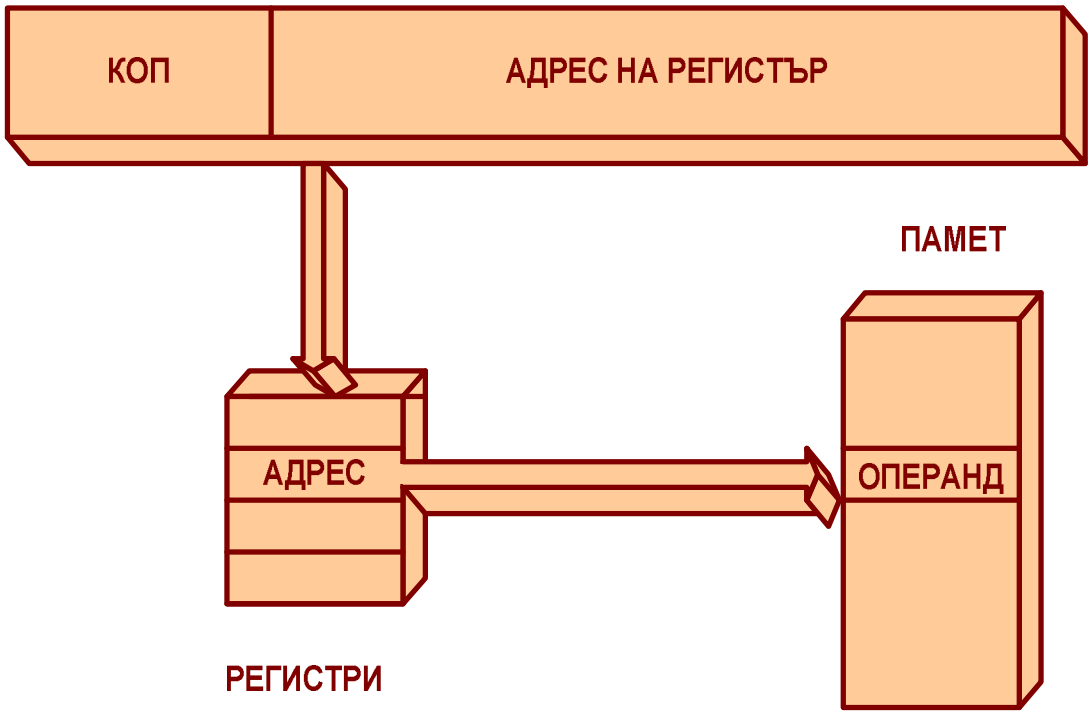


ВЛОЖЕНО АДРЕСИРАНЕ

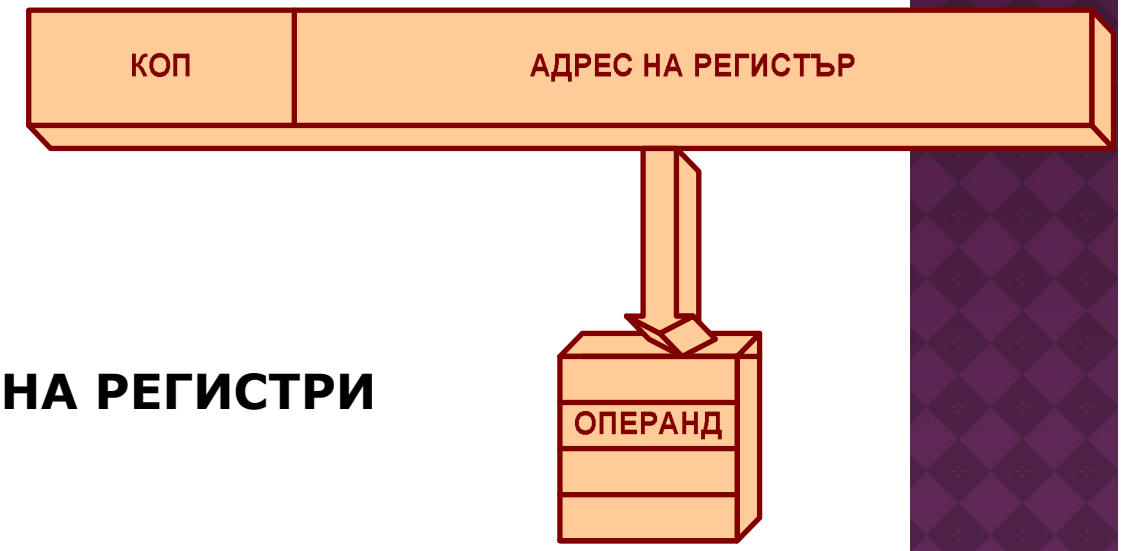
ДИРЕКТНО АДРЕСИРАНЕ

ИНДИРЕКТНА АДРЕСАЦИЯ



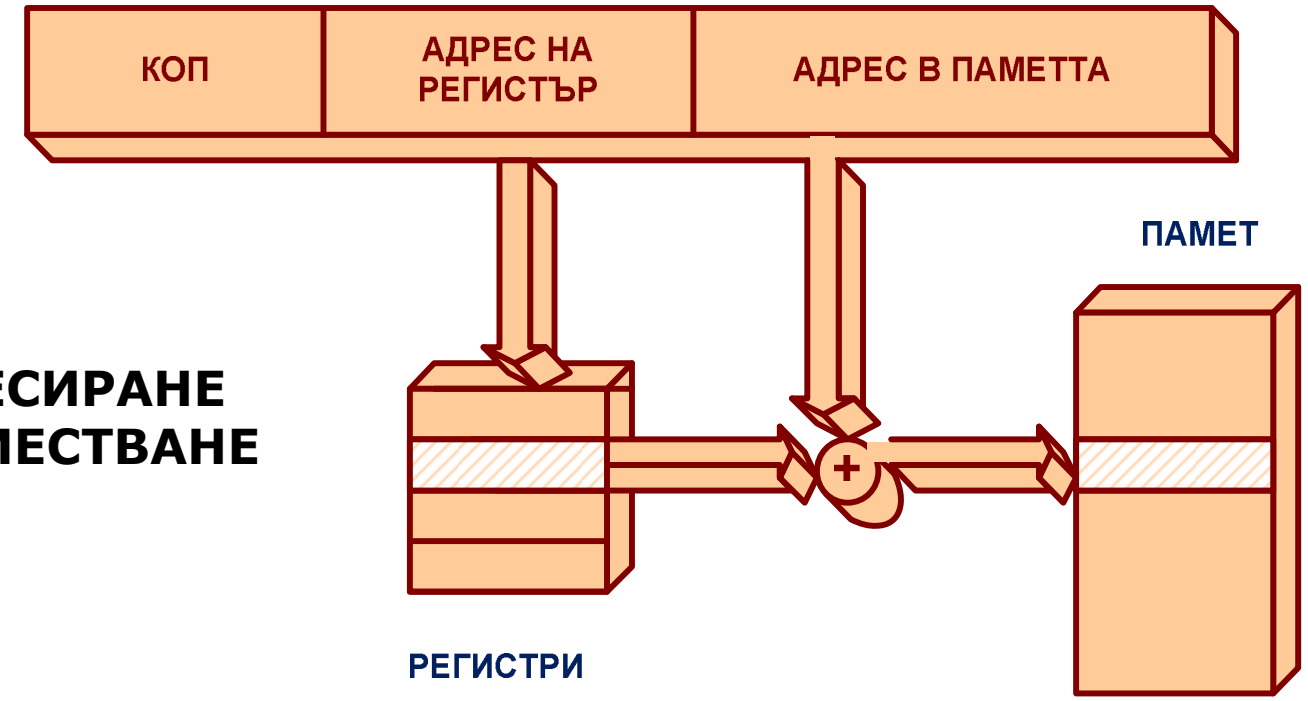


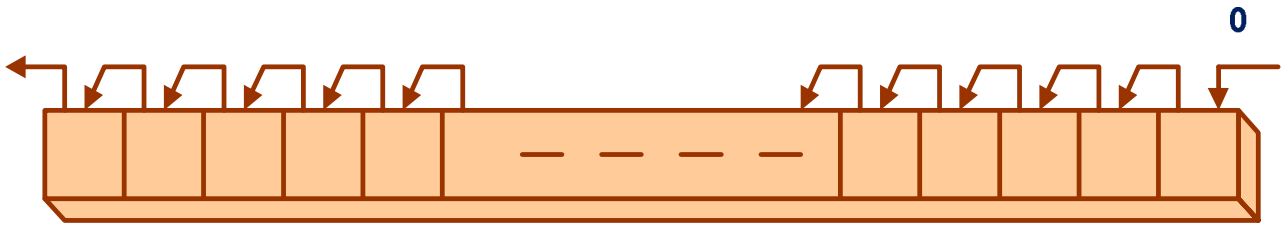
ИНДИРЕКТНО АДРЕСИРАНЕ С РЕГИСТРИ



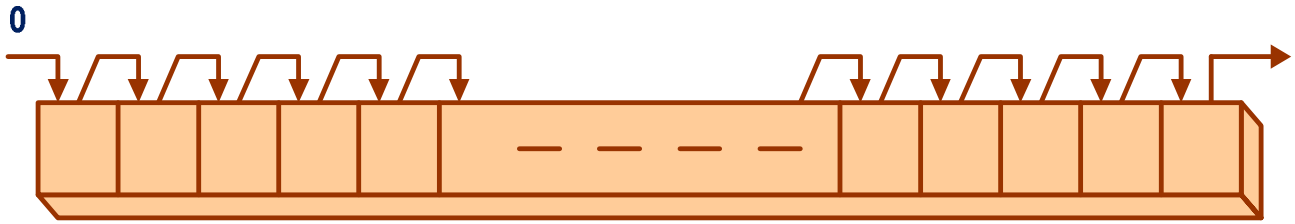
АДРЕСИРАНЕ НА РЕГИСТРИ

АДРЕСИРАНЕ С ИЗМЕСТВАНЕ

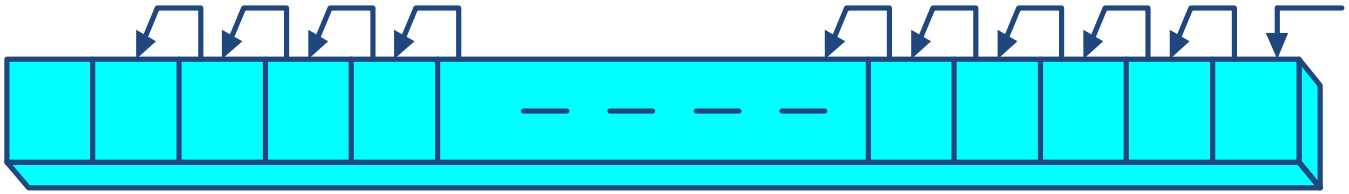




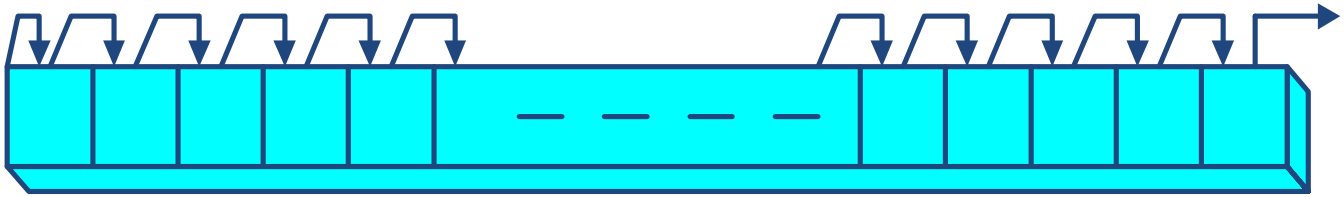
логическо изместване в ляво



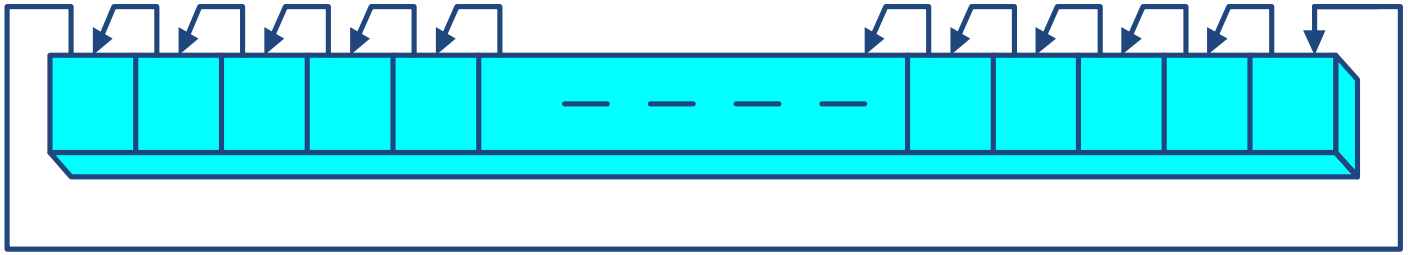
логическо изместване в дясно



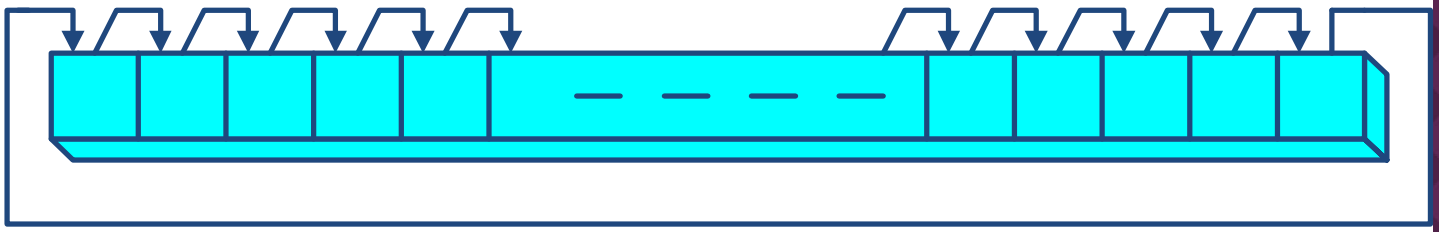
аритметично изместване в ляво



аритметично изместване в дясно



ротация в ляво (циклично изместване)



ротация в дясно (циклично изместване)

ФУНКЦИОНАЛНИ ГРУПИ ИНСТРУКЦИИ

- ИНСТРУКЦИИ ЗА ПРЕДАВАНЕ НА УПРАВЛЕНИЕТО
- ИНСТРУКЦИИ ЗА ТРАНСФЕР
- ИНСТРУКЦИИ ЗА АРИТМЕТИКО-ЛОГИЧЕСКА ОБРАБОТКА
- ИНСТРУКЦИИ ЗА СИМВОЛНА ОБРАБОТКА
- ИНСТРУКЦИИ ЗА ВХОД/ИЗХОД
- ИНСТРУКЦИИ ЗА MMX ТЕХНОЛОГИЯ И ТЕХНОЛОГИЯ С ПОТОКОВИ SIMD РАЗШИРЕНИЯ
- ИНСТРУКЦИИТЕ ЗА УПРАВЛЕНИЕ НА СИСТЕМАТА - ПРИВИЛЕГИРОВАНИ ИНСТРУКЦИИ
- ИНСТРУКЦИИТЕ ЗА КОНВЕРТИРАНЕ НА ДАННИТЕ

ВИДОВЕ ОПЕРАЦИИ

- M-M (Memory-Memory) – памет – памет най-бавните операции, операндите и резултатът са в паметта, най-голям инструкционен формат, компактно представяне на програмата (мин. памет)
- R-R (Register-Register) – регистър – регистър най-бързите операции, изискват допълнителни операции load (за зареждане на операндите в регистрите) и store (запис на резултата в паметта), най-малък инструкционен формат, представяне на програмата (макс. памет)
- M-R (Memory-Register) - памет – регистър – междинно място

ОРТОГОНАЛНА СИСТЕМА ИНСТРУКЦИИ

- При ортогоналната система инструкции инструкцията не е обвързана с определени режими на адресация т.е. всяка инструкция може да се изпълни с всеки режим на адресация (addressing mode)
- При микропроцесорите на Intel системата инструкции не е ортогонална т.е. за всяка инструкция е дефинирано допустимо множество режими на адресация

INTEL X86 ARCHITECTURE

- Intel x86 architecture поддържа инструкции с различна дължина, преобладаващо двуадресни инструкции от типа "CISC" фокусиран върху програмна съвместимост с по-старите модели.
- Системата инструкции, обаче, не е точно CISC, а по своята същност представлява разширена версия на простите 8-битови 8008 и 8080 архитектури.
- Поддържа се адресирането на байтове
- Машинните думи се съхраняват в паметта в *реда "little-endian byte order"*.
- Поддържа достъпи в паметта до неподравнение адреси за всички размери на машинната дума.
- Типичните инструкции са с дължина 2 или 3 bytes (някои инструкции са много по-дълги, други са еднобайтови).

INTEL 64 AND IA-32 ARCHITECTURES INSTRUCTION ENCODINGS

- Процесорите Intel® CoreTM i7 и Intel® CoreTM i5 са базирани на Intel® microarchitecture code name Nehalem и поддържат Intel 64 architecture.
- Процесорите, базирани на Intel® microarchitecture code name Westmere поддържат Intel 64 architecture.
- P6 family, Pentium® M, Intel® Core™ Solo, Intel® Core™ Duo processors, dual-core Intel® Xeon® processor LV, както и по-старите генерации Pentium 4 и Intel Xeon поддържат IA-32 architecture.
- Intel® AtomTM processor Z5xx series поддържа IA-32 architecture.
- Фамилиите процесори Intel® Xeon® processor E7-8800/4800/2800, Intel® Xeon® processor 3000, 3200, 5000, 5100, 5200, 5300, 5400, 7100, 7200, 7300, 7400 series, Intel® Core™2 Duo, Intel® Core™2 Extreme, Intel® Core™2 Quad processors, Pentium® D processors, Pentium® Dual-Core processor, новите генерации процесорни фамилии (processor family) Pentium 4 и Intel Xeon поддържат Intel® 64 architecture.
- IA-32 architecture е ISA (instruction set architecture) и програмна среда (programming environment) за Intel 32-bit микропроцесори.
- Intel® *64 architecture е* ISA (*instruction set architecture)* и програмна среда *(programming environment)*, която представлява разширено множество (superset) на Intel 64-bit архитектури. Съвместима е с IA-32 architecture.

IPL АРХИТЕКТУРИ

IPL АРХИТЕКТУРИ	ГРУПИРАНЕ	ПЛАНИРАНЕ НА ФУНКЦИОНАЛНИТЕ УСТРОЙСТВА	ИНИЦИИРАНЕ
СУПЕРСКАЛАРНИ	ОТ ХАРДУЕРА	ОТ ХАРДУЕРА	ОТ ХАРДУЕРА
ЕКСПЛИЦИТНО ПАРАЛЕЛНО ИЗПЪЛНЕНИЕ НА ИНСТРУКЦИИ EPIC	ОТ КОМПИЛАТОРА	ОТ ХАРДУЕРА	ОТ ХАРДУЕРА
ДИНАМИЧНИ АРХИТЕКТУРИ С МНОГО ДЪЛГИ ИНСТРУКЦИОННИ ДУМИ (Dynamic VLIW)	ОТ КОМПИЛАТОРА	ОТ КОМПИЛАТОРА	ОТ ХАРДУЕРА
АРХИТЕКТУРИ С МНОГО ДЪЛГИ ИНСТРУКЦИОННИ ДУМИ (VLIW)	ОТ КОМПИЛАТОРА	ОТ КОМПИЛАТОРА	ОТ КОМПИЛАТОРА

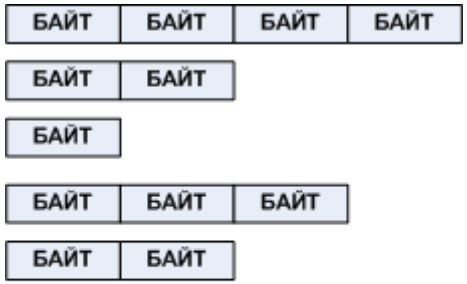
VLIW ARCHITECTURES
EPIC

- VLIW - Very Long Instruction Word
- EPIC – Explicitly Parallel Instruction Computing
- При типичната VLIW архитектура дължината на инструкциите е стотици битове.
- В рамките на VLIW процесор функционират паралелно множество функционални устройства.
- Всички функционални устройства споделят огромен общ регистров файл.

АРХИТЕКТУРНО СРАВНЕНИЕ

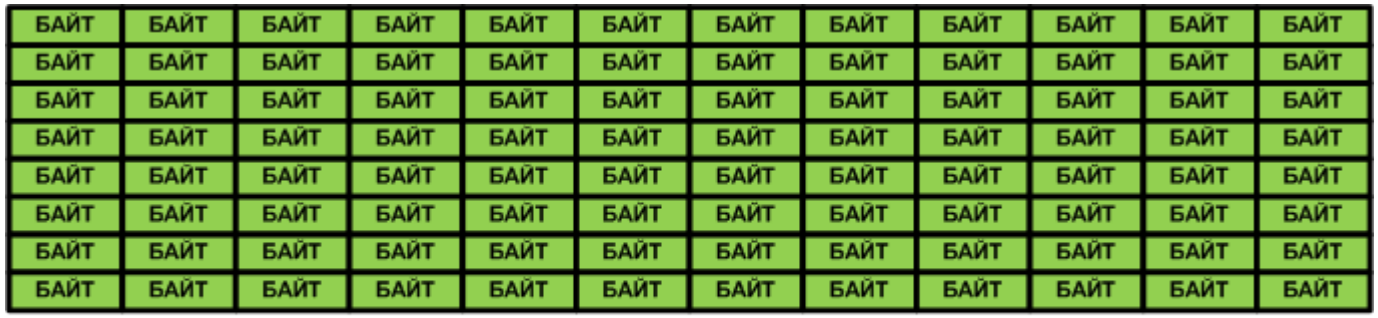
АРХИТЕКТУРНИ ХАРАКТЕРИСТИКИ	CISC	RISC	VLIW
ДЪЛЖИНА НА ИНСТРУКЦИЯТА	ПРОМЕНЛИВА	ЕДНАКВА, ОБИКНОВЕНО 32 БИТА	ЕДНАКВА
ФОРМАТ НА ИНСТРУКЦИЯТА	РАЗПОЛОЖЕНИЕТО НА ПОЛЕТАТА СЕ ПРОМЕНЯ	РЕГУЛЯРНО, РАЗПОЛОЖЕНИЕТО НА ПОЛЕТАТА НЕ СЕ ПРОМЕНЯ	РЕГУЛЯРНО, РАЗПОЛОЖЕНИЕТО НА ПОЛЕТАТА НЕ СЕ ПРОМЕНЯ
СЕМАНТИКА НА ИНСТРУКЦИИТЕ	ПРОМЕНЯ СЕ ОТ ПРОСТА ДО СЛОЖНА	САМО ЕДНА ОПЕРАЦИЯ	МНОЖЕСТВО ПРОСТИ, НЕЗАВИСИМИ ОПЕРАЦИИ
РЕГИСТРИ	МАЛКО, ПОНЯКОГА СПЕЦИАЛИЗИРАНИ	МНОГО, УНИВЕРСАЛНИ	МНОГО, УНИВЕРСАЛНИ
ДОСТЪПИ ДО ПАМЕТТА	ОБВЪРЗАНИ С ОПЕРАЦИИ НА РАЗЛИЧНИ ТИПОВЕ ИНСТРУКЦИИ	НЕ СА ОБВЪРЗАНИ С ОПЕРАЦИИ, LOAD/STORE АРХИТЕКТУРИ	НЕ СА ОБВЪРЗАНИ С ОПЕРАЦИИ, LOAD/STORE АРХИТЕКТУРИ
АКЦЕНТ ПРИ ПРОЕКТИРАНЕТО НА ХАРДУЕРА	ИЗПОЛЗВАНЕ НА МИКРОКОДИРАНИ ИМПЛЕМЕНТАЦИИ	ИМПЛЕМЕНТАЦИИ С ЕДИН КОНВЕЙЕР И БЕЗ МИКРОКОД	ИМПЛЕМЕНТАЦИИ С МНОЖЕСТВО КОНВЕЙЕРИ, БЕЗ МИКРОКОД И БЕЗ СЛОЖНА ЛОГИКА ЗА ДИСПЕЧЕРИЗАЦИЯ

ИЗПЪЛНЕНИЕ НА ТИПИЧНИ ИНСТРУКЦИИ



CISC

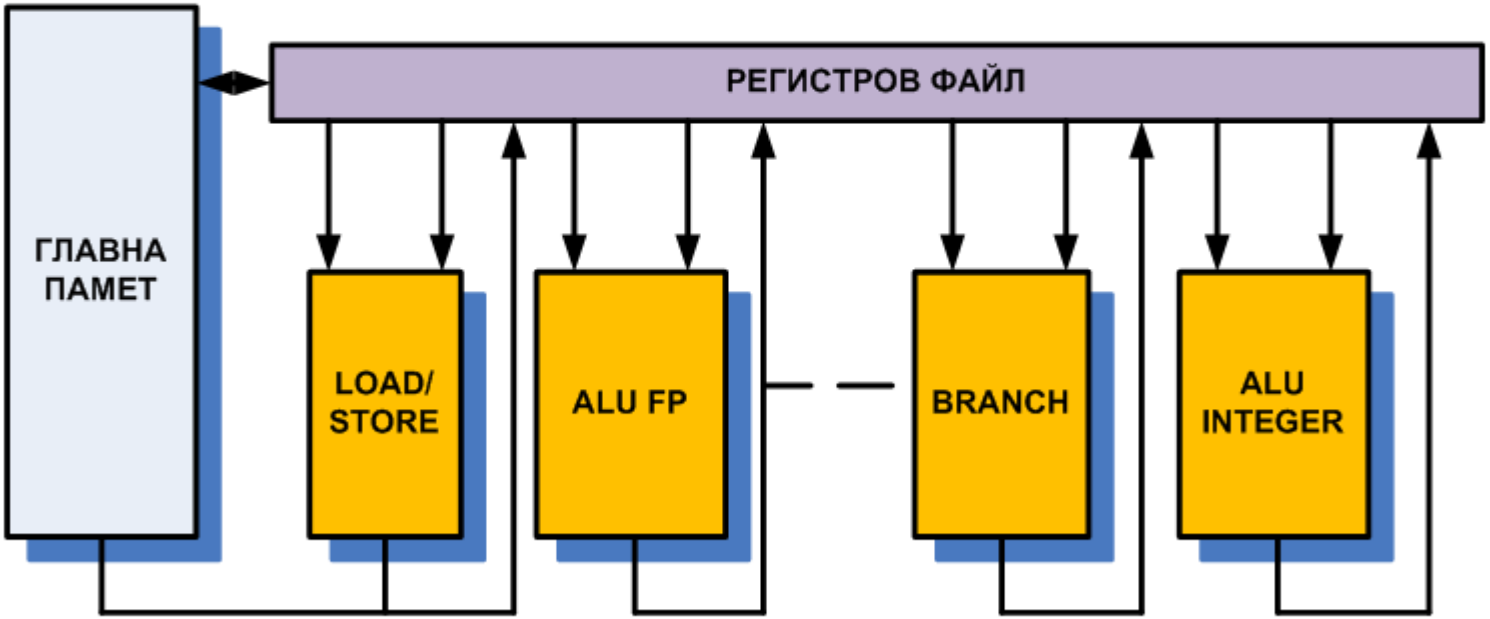
RISC



VLIW

АРХИТЕКТУРА НА VLIW ПРОЦЕСОР

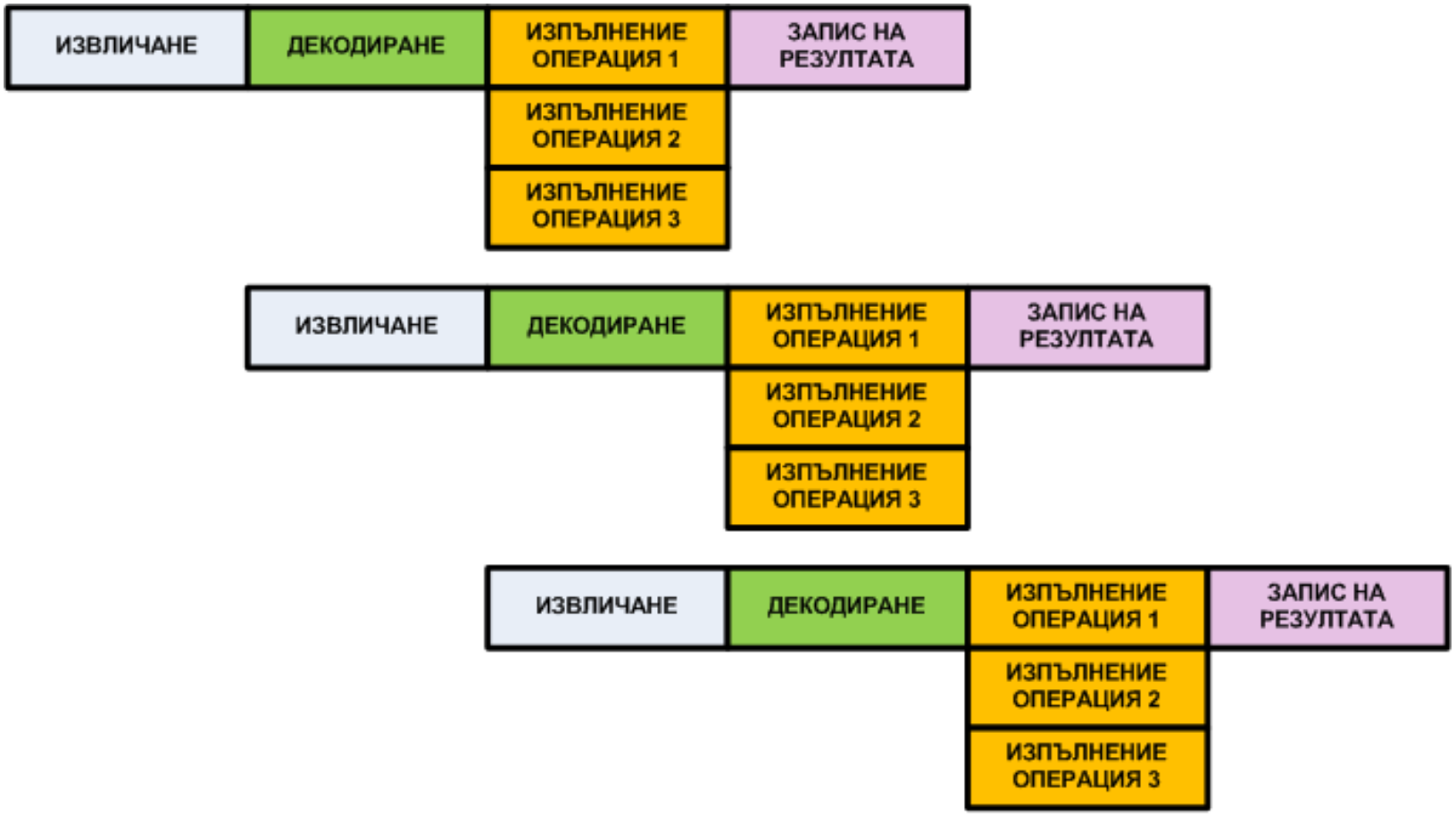
ФОРМАТ НА ИНСТРУКЦИЯТА ПРИ VLIW АРХИТЕКТУРА



АРХИТЕКТУРА НА VLIW ПРОЦЕСОР

ПРОЦЕСОРИ VLIW

spatial parallelism



КОНВЕЙЕРНО ИЗПЪЛНЕНИЕ НА ИНСТРУКЦИИТЕ
ПРИ VLIW ПРОЦЕСОР СЪС СТЕПЕН 3

ПРЕДИМСТВА НА VLIW ПРОЦЕСОРИТЕ

Компилаторите генерират фиксирани инструкционни пакети (instruction packets) с множество операции (в зависимост от степента на конвейера) и създават плана за изпълнението им

- Зависимостите между инструкциите се определят от компилатора и се използват при планирането в съответствие със латентностите на функционалните устройства
- Функционалните устройства се планират от компилатора според позицията в инструкционните пакети ("slotting")
- Компилаторът генерира напълно планиран за изпълнение код, в който липсват хазарти (fully-scheduled, hazard-free code) => не се налага хардуерът да открива зависимости или да планира

НЕДОСТАТЪЦИ НА VLIW ПРОЦЕСОРИТЕ

- Основен проблем е съвместимостта (Compatibility) на различните имплементации
  - Имплементациите са машинно-зависими - VLIW код няма да се изпълнява коректно при различен брой на функционалните устройства и различни латентности
  - Непланирани събития (напр., липса в кеша) блокират целия процесор
- Друг проблем е плътността на кода (Code density)
  - Недостатъчна утилизация на слота (предимно nops)
  - Броят на nops може да се намали чрез компресиране ("flexible VLIW", "variable-length VLIW")

КРАЙ