



Анализ и оптимизация на софтуерни приложения

Александър Пенев

Васил Василев

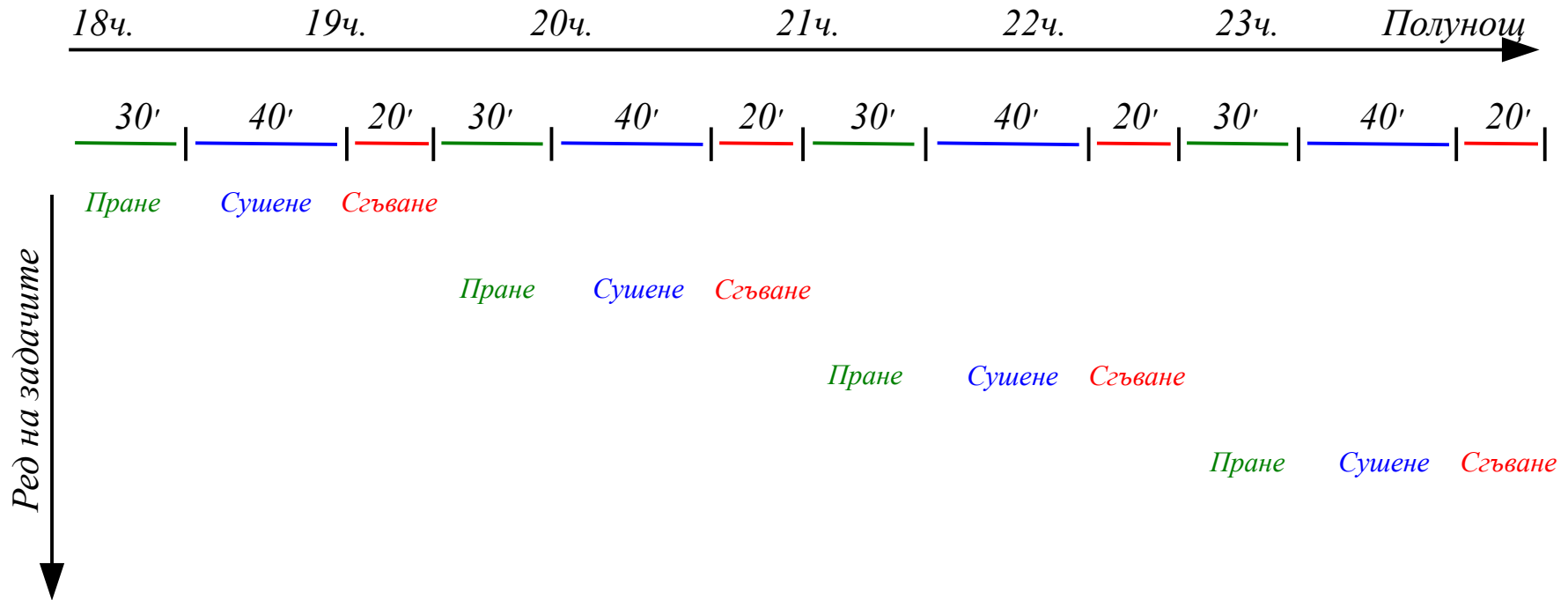
Микроархитектури

Съдържание

- ❖ Конвейерно изпълнение
- ❖ Проблеми при конвейерното изпълнение
- ❖ Структурни опасности
- ❖ Даннови опасности
- ❖ Контролни опасности
- ❖ SIMD
- ❖ Предсказване на преходи и спекулативно изпълнение

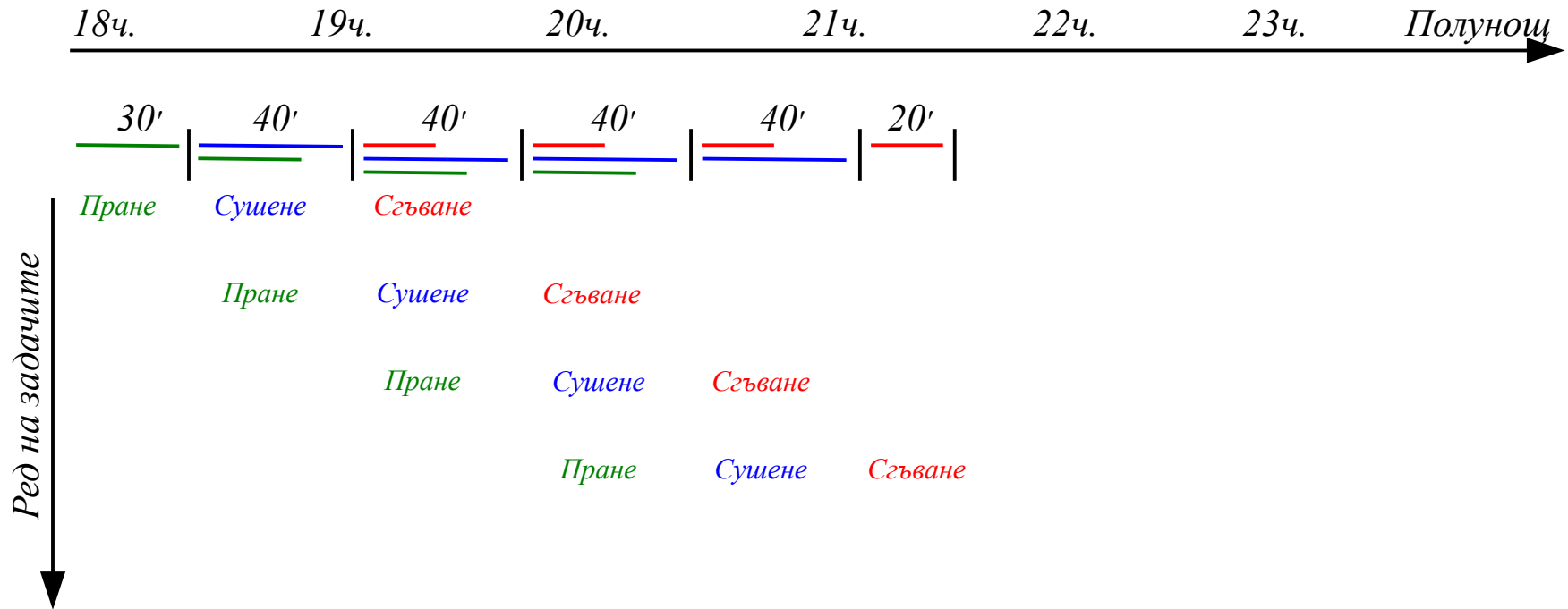


Последователно пране



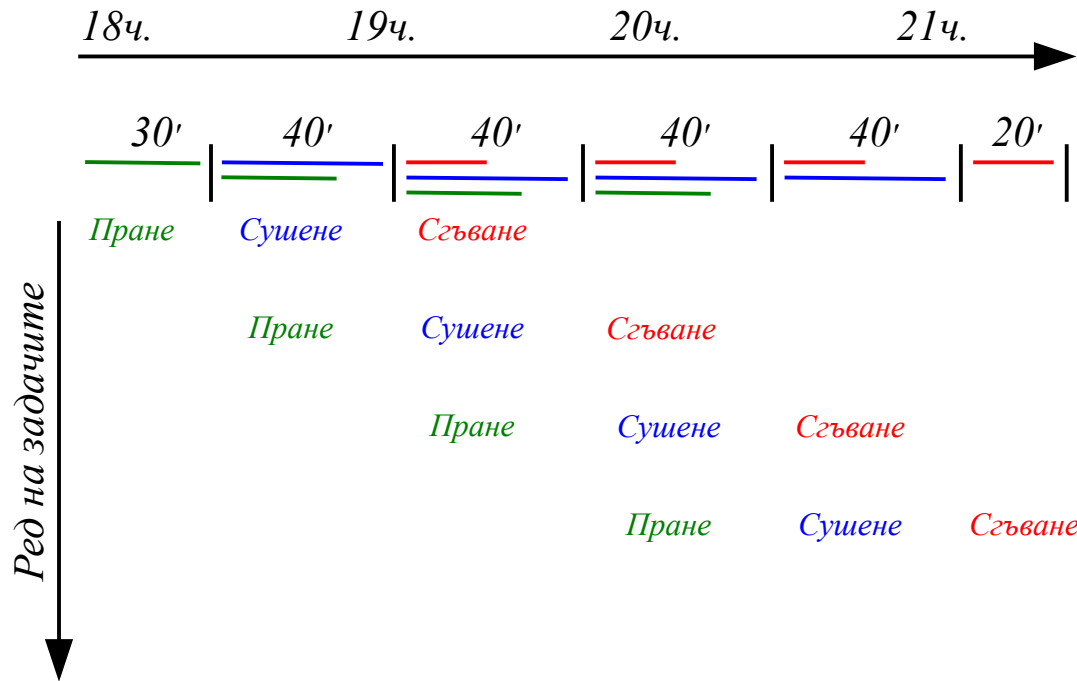
- ❖ Прането отнема 30 мин
- ❖ Сушенето отнема 40 мин
- ❖ Сгъването отнема 20 мин
- ❖ Последователното пране отнема 6 часа за 4 купа дрехи

Конвейерно пране



- ❖ Конвейерно означава задачата да се започва колкото се може по-скоро
- ❖ Конвейерното пране отнема 3.5 часа!

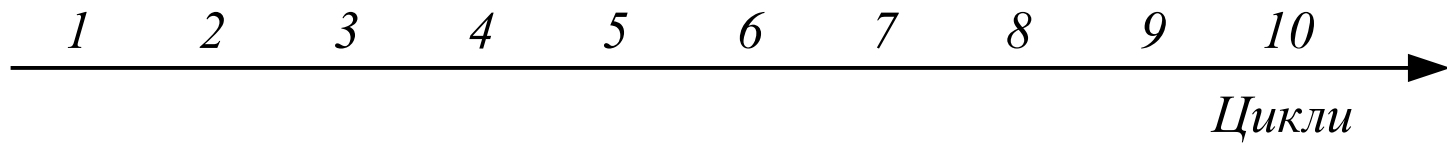
Изводи от конвейерното изпълнение



- ❖ Множествено задачи се изпълняват едновременно когато използват различни ресурси
- ❖ Възможното ускорение е броя стъпки в конвейера
- ❖ Небалансираните дължини на стъпките намаляват ускорението
- ❖ Времето за напълване на конвейера и времето на изпразване намаляват ускорението
- ❖ Stall при наличие на зависимости

- ❖ Конвейерното изпълнение не намалява латентността на отделните задачи, а само пропускателната способност на цялото задание
- ❖ Ефективността на конвейера е ограничена от най-бавната операция

Изпълнение на инструкции



Инструкция №

Цикли

Инструкция i

Инструкция i+1

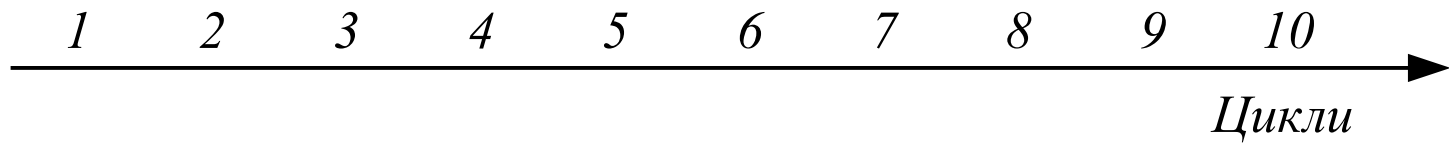
Инструкция i+2

Инструкция i+3

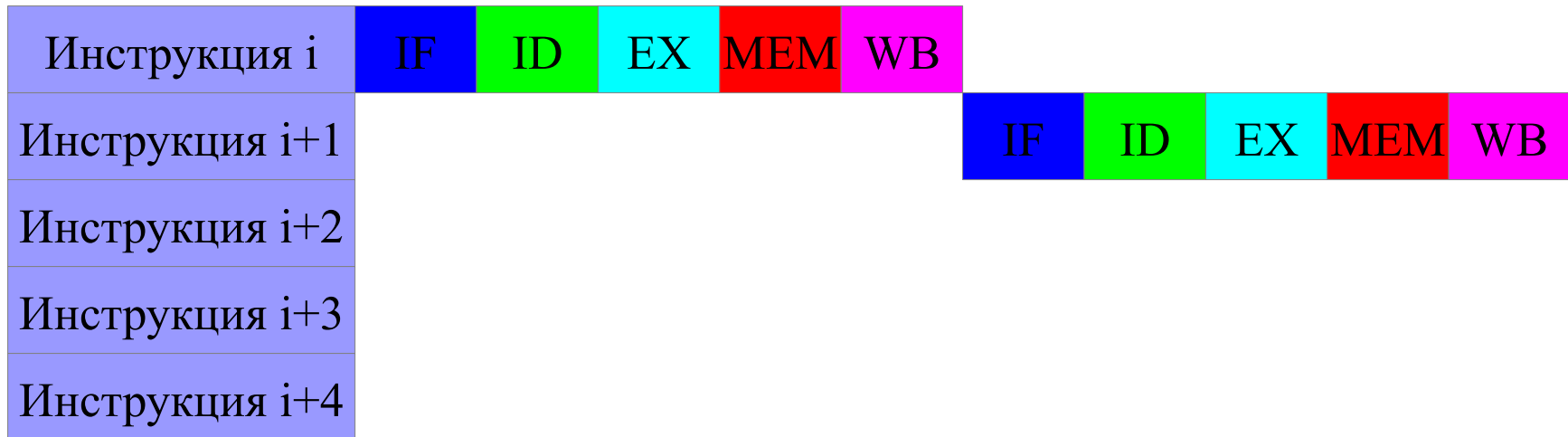
Инструкция i+4



Изпълнение на инструкции



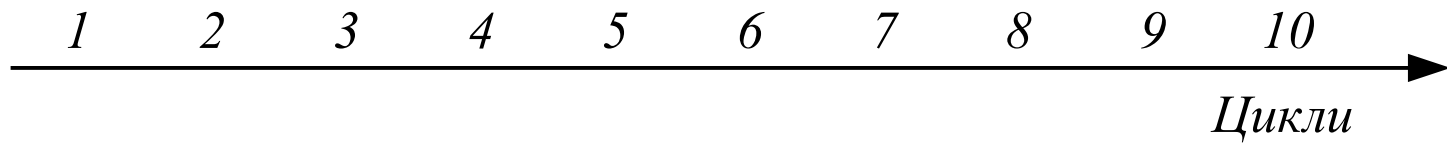
Инструкция №



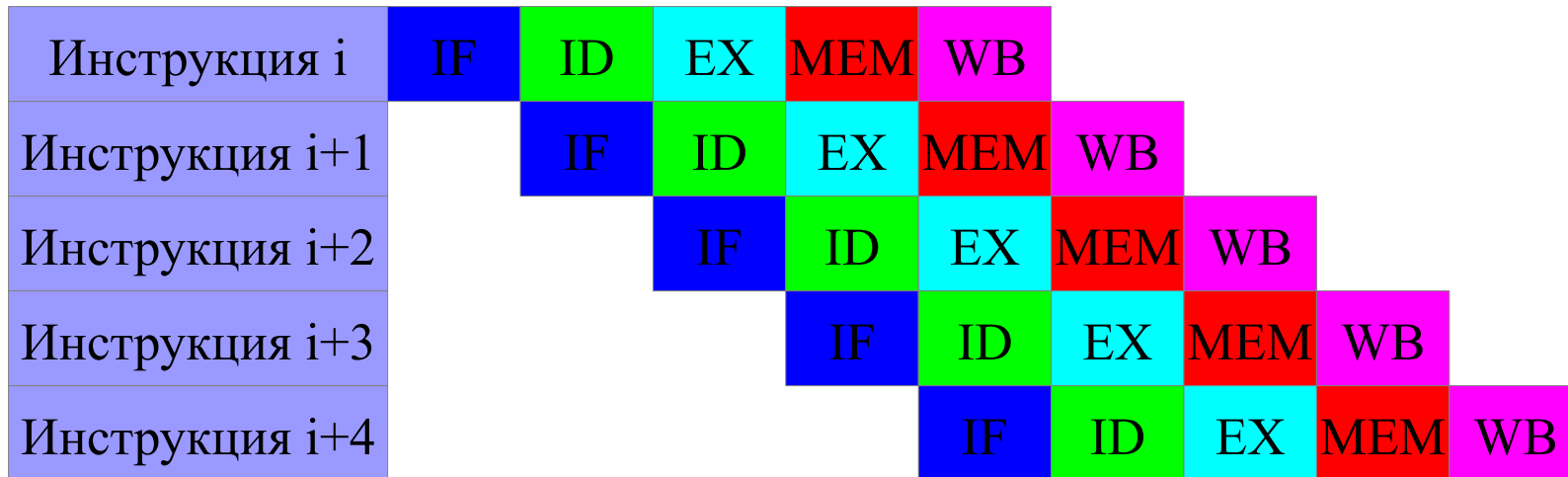
- ❖ IF – Instruction Fetch
- ❖ ID – Instruction Decode
- ❖ EX – Execution
- ❖ MEM – Memory Access
- ❖ WB – Write Back



Изпълнение на инструкции



Инструкция №



- ❖ IF – Instruction Fetch
- ❖ ID – Instruction Decode
- ❖ EX – Execution
- ❖ MEM – Memory Access
- ❖ WB – Write Back



Ограничения при конвейерното изпълнение

Опасностите (hazards) не позволяват следващата инструкция да бъде изпълнена по време на определения ѝ цикъл

- ❖ Структурни опасности (structural hazards)

Опити за използване на един и същ хардуер за две различни неща едновременно

- ❖ Даннови опасности (data hazards)

Инструкция зависи от резултата на друга инструкция, която е още в конвейера

- ❖ Контролни опасности (control hazards)

Причинени са от забавянето между извличането на инструкции и решенията за промени в потока инструкции (control flow) – условни и безусловни преходи



Даннови опасности: без зависимост

Инструкция 3 не зависи от данните, получени от инструкция 2. В двете инструкции се четат само от данните получени в инструкция 1.

```
1.  A = 3
2.  B = A
3.  C = A
```

Ако две инструкции не са взаимнозависими от данните си те обикновено могат да бъдат:

- ❖ Изпълнени едновременно
- ❖ Напълно да се застъпват в конвейера
- ❖ Да се изпълняват непоследователно (out-of-order)

При настъпване на такава ситуация в конвейера се казва, че имаме четене след четене (RAR), което не е риск



Даннови опасности: истинска зависимост

Инструкция 3 зависи от данните, получени от предходната инструкция

```
1.  A = 3
2.  B = A
3.  C = B
```

Ако две инструкции са взаимнозависими от данните си те не могат да бъдат:

- ❖ Изпълнени едновременно
- ❖ Напълно да се застъпват в конвейера
- ❖ Да се изпълняват непоследователно (out-of-order)

При настъпване на такава ситуация в конвейера се казва, че има риск от четене след запис (RAW hazard)

Даннови опасности: анти зависимост

Инструкция 3 зависи от
инструкция 2 (инстр. 2 трябва
да се изпълни преди инстр. 3)

```
1.  B = 3
2.  A = B + 1
3.  B = 7
```

Нарича се още зависимост от имената

Когато две инструкции използват един и същи регистър или адрес в паметта, но няма поток от данни между тези две инструкции. Има две версии на този тип зависимост

При настъпване на такава ситуация в конвейера се казва, че има риск от запис след четене (WAR hazard)



Даннови опасности: изходна зависимост

Инструкция 3 зависи от
инструкция 1 (инстр. 1 трябва
да се изпълни преди инстр. 3)

```
1.  A = 2 * X
2.  B = A / 3
3.  A = 9 * Y
```

Нарича се още зависимост от имената

Когато две инструкции използват един и същи регистър или адрес в паметта, но няма поток от данни между тези две инструкции. Има две версии на този тип зависимост

При настъпване на такава ситуация в конвейера се казва, че има риск от запис след запис (WAW hazard)



Зависимост от имената

Зависимостта от имената може да се избегне като се смени името в инструкцията, така че да няма конфликт

- ❖ Софтуерно преименуване (на регистри)

Прави се от компилатора

- ❖ Хардуерно преименуване (на регистри)

Прави се от процесора. Недостатък: по време на изпълнение

```
1. B = 3
2. A = B + 1
3. B = 7
```

```
1. B = 3
2. B2 = B
3. A = B2 + 1
4. B = 7
```

WAR опасност

```
1. A = 2 * X
2. B = A / 3
3. A = 9 * Y
```

```
1. A2 = 2 * X
2. B = A2 / 3
3. A = 9 * Y
```

WAW опасност



Контролни опасности

Всяка инструкция е зависима от някакво множество преходи. Тези зависимости трябва да се спазват, за да може да се запази семантиката на програмата

```
if c1 {  
    I1;  
}  
if c2 {  
    I2;  
}
```

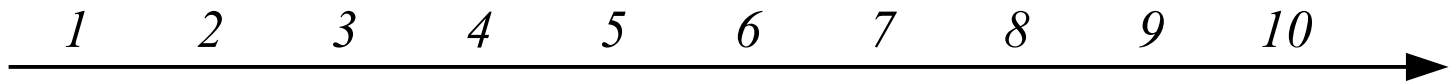
I1 зависи от условието c1, а I2 зависи от условието c2, но не и от c1.

Контролни опасности

Избягването на тези конфликти е трудно. Единственият начин е да не се спазва последователността от изпълнение. С други думи изпълнение на инструкции, които не би трябвало да се изпълняват и следователно водят до нарушаване на контролните зависимости. Това е приложимо само когато може да се запази семантиката на програмата. Този вид изпълнение се нарича спекулативно изпълнение.



Суперскаларно изпълнение



Тип на инструкция

Integer	IF	ID	EX	MEM	WB					
Floating point	IF	ID	EX	MEM	WB					
Integer		IF	ID	EX	MEM	WB				
Floating point		IF	ID	EX	MEM	WB				
Integer			IF	ID	EX	MEM	WB			
Floating point			IF	ID	EX	MEM	WB			
Integer				IF	ID	EX	MEM	WB		
Floating point				IF	ID	EX	MEM	WB		
Integer					IF	ID	EX	MEM	WB	
Floating point					IF	ID	EX	MEM	WB	

Двоен суперскалар



Даннови опасности при ILP

- ❖ Локализация на Instruction level parallelism

Много инструкции да се изпълняват паралелно

- ❖ Хардуерът/Софтуерът трябва да запази редът на изпълнение на програмите

Трябва да се запази редът на изпълнение на инструкциите и да изглежда така че те се изпълняват от последователно и следват сорс кода

- ❖ Важност на данновите зависимости:

- ❖ Отбелязва възможни опасности, а не задължителни проблеми
- ❖ Определя редът, в който резултатите трябва да се калкулират
- ❖ Ограничава отгоре колко паралелизъм може да се използва

Крайна цел: използване на паралелизма като се запази последователното изпълнение само когато то има негативен ефект върху изпълнението на програмата



Мултимедийни инструкции

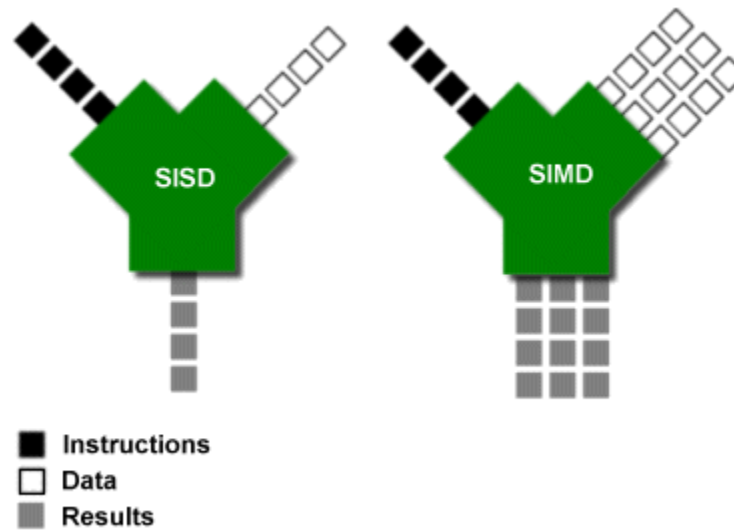
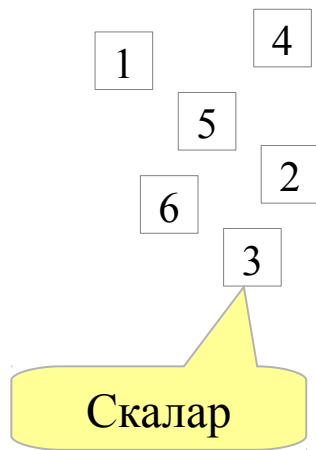
- ❖ SIMD (Single Instruction Multiple Data)

Техниката се използва, за да се постигне паралелизъм на ниво данни като при векторните процесори

- ❖ Една от най-новите реализации на SIMD от Интел се нарича SSE



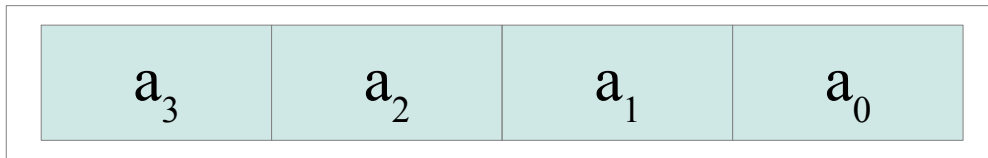
Мултимедийни инструкции



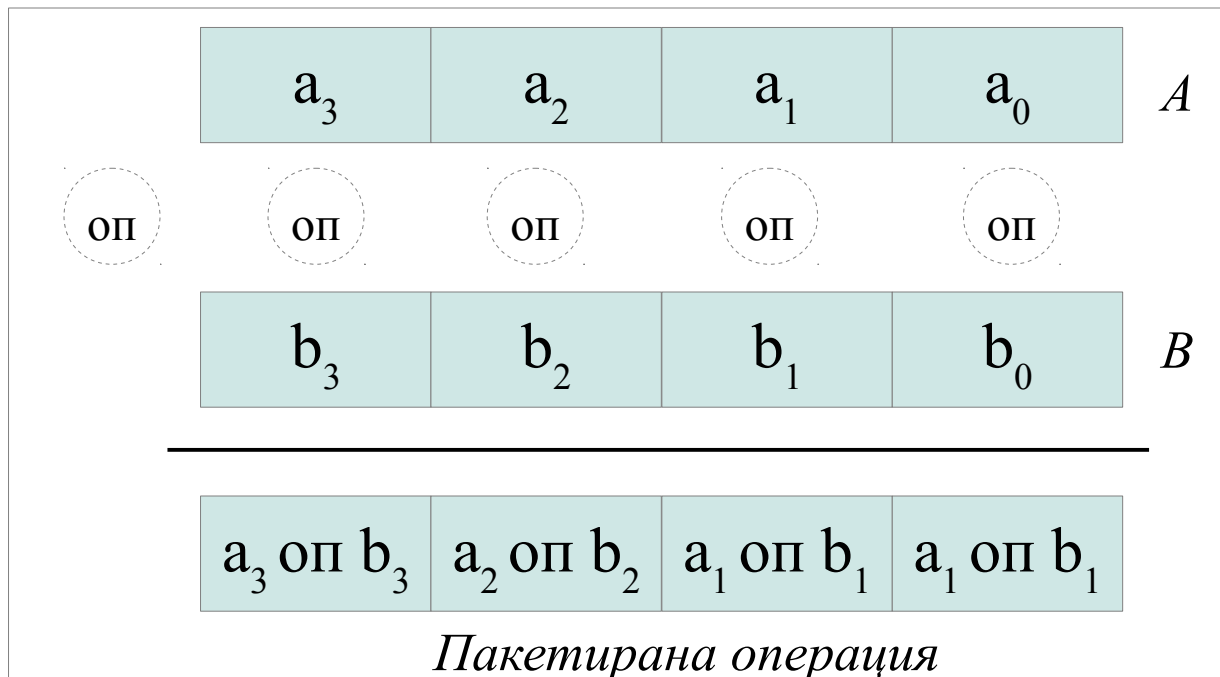
Мултимедийни инструкции

❖ Пакетиран тип данни

Намират се в отделен регистров файл



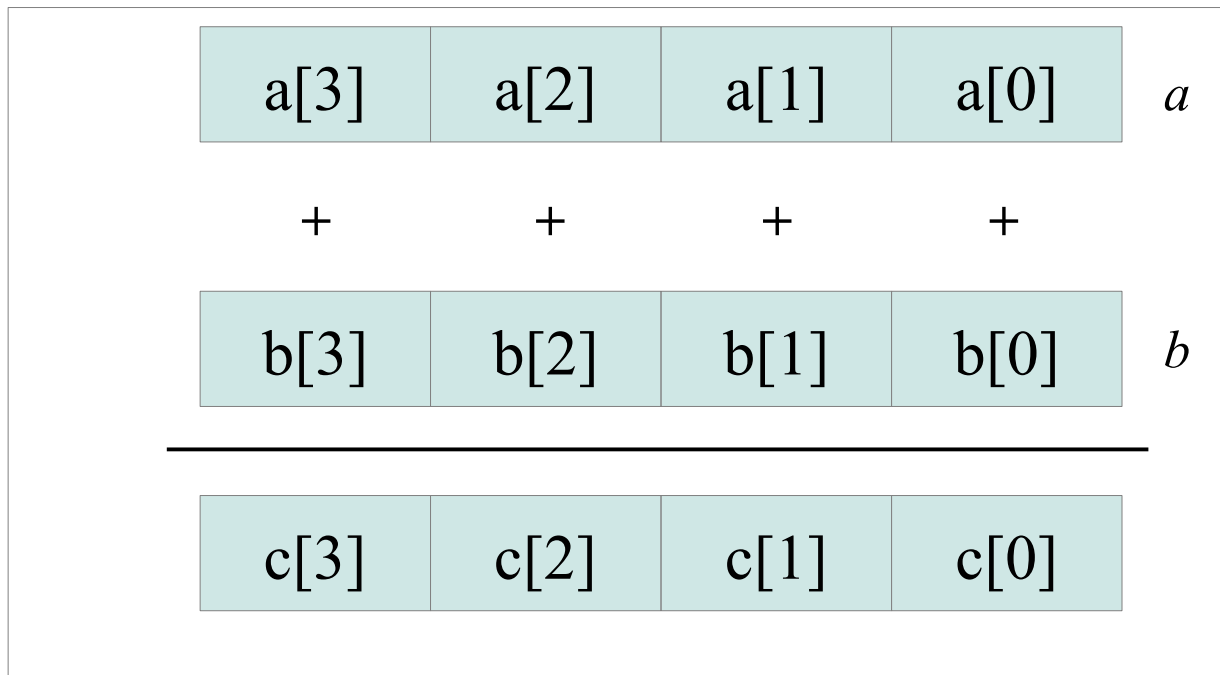
❖ SIMD



Мултимедийни инструкции

```
for (i=0; i < MAX; ++i)  
    c[i] = a[i] + b[i];
```

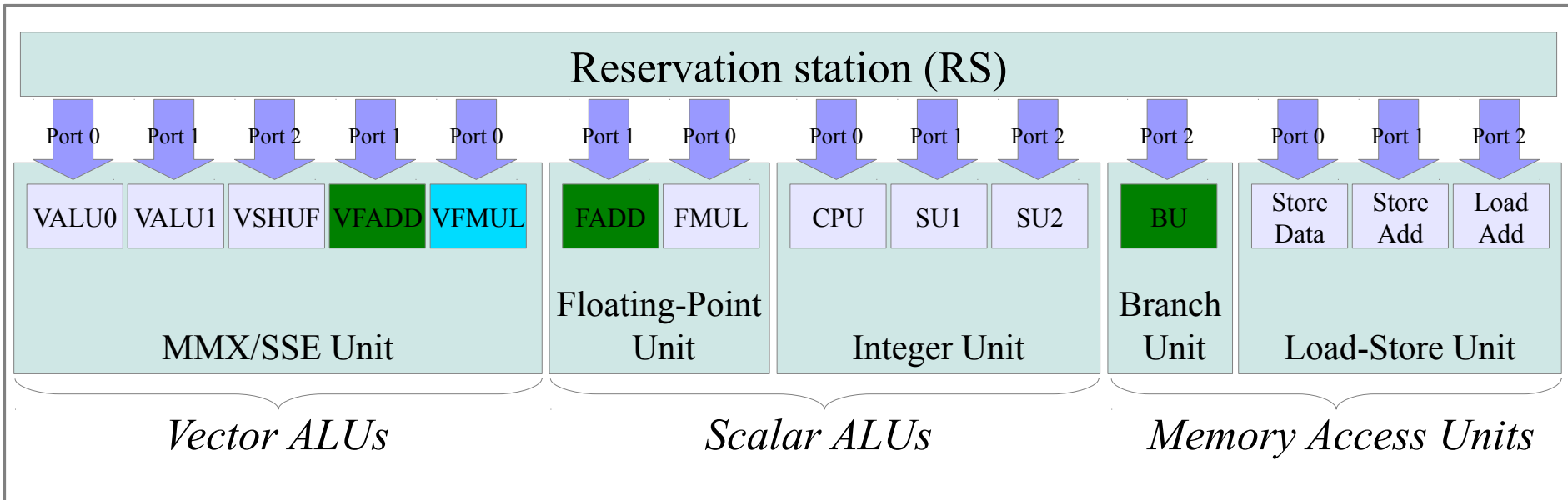
❖ SIMD



Intel Core Микроархитектура.

Суперскаларно изпълнение

- ❖ 3 Load/Store
- ❖ 1 Branch
- ❖ 3 Integer
- ❖ 2 Floating point
- ❖ 5 SSE



Спекулативно изпълнение

Различни механизми за предсказване

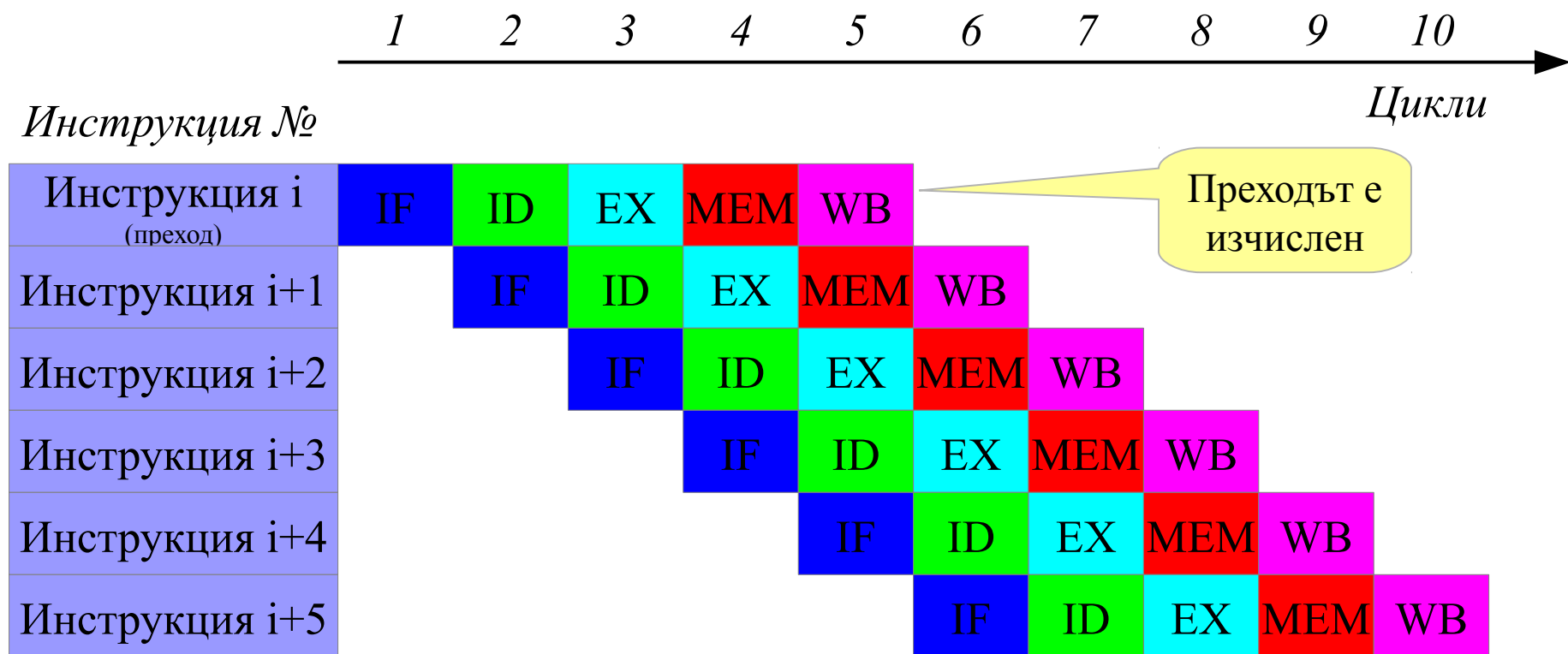
Постига се по-добра ILP като се преодоляват контролните зависимости чрез спекулиране на резултата от преходите и изпълнява програмата сякаш предсказанията са верни

- ❖ Предсказване на преходи
- ❖ Предсказване на стойности
- ❖ Prefetching

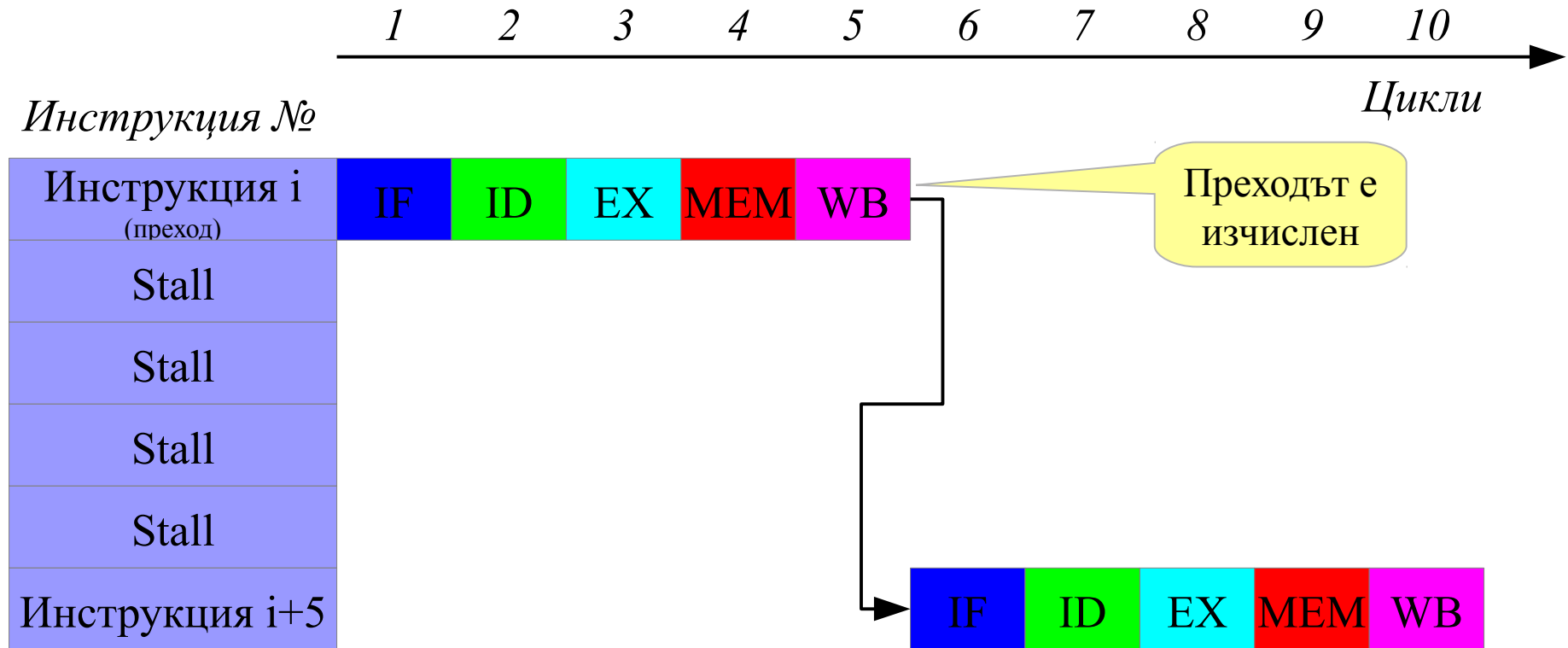
Предвиждане на начинът на достъп до паметта



Предсказване на преходи и спекулативно изпълнение



Предсказване на преходи и спекулативно изпълнение



Предсказване на преходи и спекулативно изпълнение

Механизмът за предсказване трябва да определи кое от двете разклонения на условието ще се изпълни

- ❖ Съвременните предсказващи системи са 99+% точни

Дори могат да предсказват адресите от индиректните преходи

Извличат се и се изпълняват спекулативно предсказаните адреси

- ❖ Няма мехурчета в конвейера

Когато преходът най-накрая е оценен и адресът е ясен тогава спекулативното изпълнение се потвърждава или отхвърля

