Състезания за управление при конвейерно изпълнение

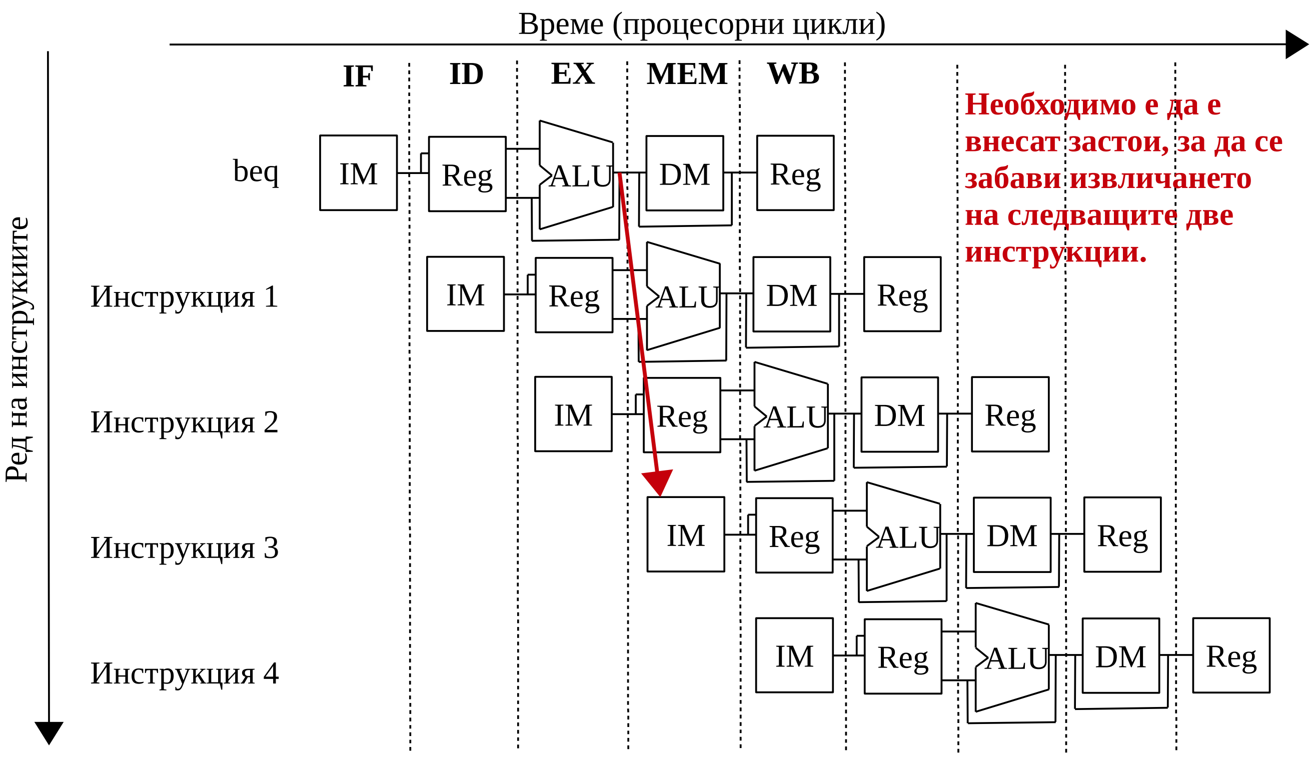
Съдържание:

1. Състезания за управление. Внасяне на застои;
2. Намаляване на застоите при изпълнение на инструкции за преход;
3. Стратегии за предсказване на преходи.

1. Състезания за управление. Начини за преодоляването им

В предишната лекция разгледахме структурните състезания и зависимостите по данни. Посочихме и различни техники за тяхното преодоляване. Лекцията за тази седмица е посветена на състезанията за управление (на последователността на изпълнението на инструкциите), които възникват при конвейеризирането на инструкциите за преход и други инструкции, които променят програмния брояч.

Инструкциите за преход променят последователността, в която се извличат инструкциите от паметта. Ако няма инструкции за преход от паметта винаги ще се извлича инструкцията, която се намира на следващия адрес след текущата. Вече знаете, че програмния брояч (PC), при извличането на инструкция, автоматично се увеличава с 4, тъй като всяка инструкция заема 4 клетки (байта) от паметта. При изпълнение на инструкция за преход, обаче, в програмния брояч може да се зареди адрес, който е различен от адреса на следващата инструкция. Този адрес сочи към инструкция, която може вече да е била изпълнена (връщане назад в кода на програмата) или към инструкция, която предстои да се изпълни (преход напред в кода на програмата). Проблемът тук е че, за да се стигне до зареждане на програмния броя с адрес, различен от PC + 4, инструкцията за преход трябва да премине през първите три етапа в конвейера и чак след като се изпълни на етапа EXE, програмният брояч може да се зареди с адреса, който е зададен в инструкцията за преход. До тогава PC се увеличава с 4 при всеки следващ процесорен цикъл, извлича се следващата поредна инструкция в кода на програмата и се зарежда в конвейера. По този начин докато се установи, че инструкцията за преход изисква в PC да се зареди, стойност различна от PC + 4, в конвейера вече ще са влезли две инструкции (непосредствено следващи инструкцията за преход), които ще се намират на етапи ID и IF. Описаният процес е илюстриран на фигурата по-долу.



Когато една инструкция за преход се изпълнява, тя може да промени или не *програмния брояч (PC) със стойност различна от текущата плюс 4. Ако PC бъде зареден с целевия адрес посочен в инструкцията за преход, то преходът се счита за осъществен, ако ли не – за неосъществен. Ако условието в инструкцията beq е вярно, преходът ще бъде осъществен и програмният брояч ще се промени след изчисляване на целевия адрес и тестване на условието на прехода, което се извършва от ALU в края на етапа за изпълнение на инструкциите (EXE).*

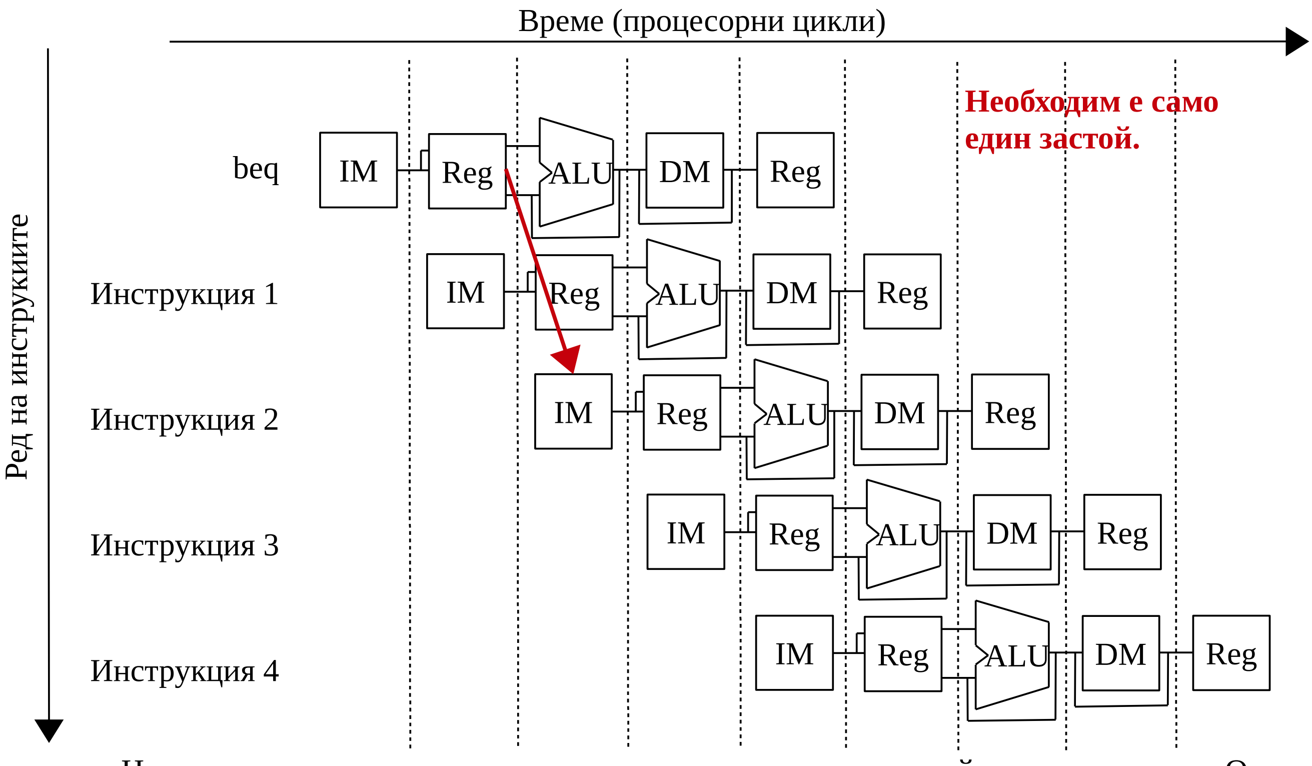
Най-простото решение, за да не започва изпълнението на инструкциите след инструкцията за преход, докато не се установи коя е следващата инструкция за изпълнение (тази посочена в инструкцията за преход, ако условието на прехода е вярно, или поредната инструкция в кода на програмата, ако условието на прехода не е вярно), е да се забави извличането на следващите инструкции с два процесорни цикъла като се внесат застои. Така всяка инструкция за преход ще внася по два процесорни цикъла забавяне в изпълнението на програмата, което е неприемливо, ако в кода има голям брой преходи.

2. Намаляване на застоите при изпълнение на инструкции за преход

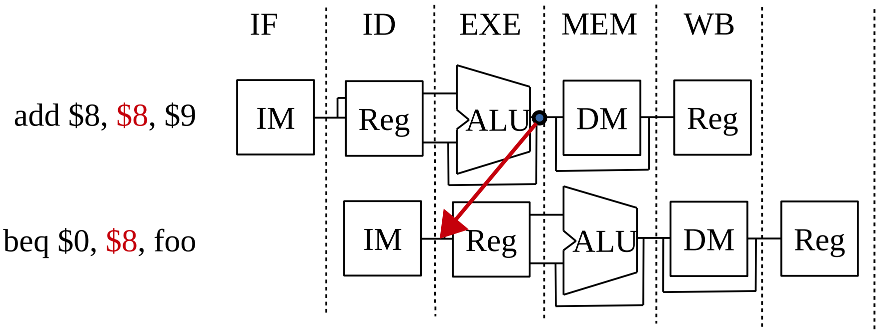
Има различни методи за справяне със застоите в конвейера, причинени от инструкции за преход. Тук ще разгледаме четири варианта прилагани по време на компилацията. Това са статични методи, т. е. за всяка инструкция за преход в конвейера се изпълняват едни и същи действия. По време на компилацията се прави опит да се минимизират забавянията в конвейера, съобразно с хардуерно заложената схема за обработка на преходи. Съществуват и хардуерно базирани схеми, които динамично прогнозират поведението на преходите. Част от тях също са разгледани.

2.1. Промяна на хардуерната схема

Изчисляването на целевия адрес и проверката на условието на прехода се премества от етапа EXE в етапа ID. Така, веднага след декодиране на инструкцията целевият адрес на прехода ще бъде наличен и ще може да се установи дали е необходимо този адрес да се зареди в програмния брояч или не. Това ще намали внасяните застои след всяка инструкция за преход от два на един, както е показано на фигурата.



Недостатък на това решение е че хардуерната схема на конвейера се усложнява. Освен това може да възникне ситуация, като показаната на следващата фигура, за разрешаването на която ще е необходимо добавяне на още хардуерни ресурси в конвейера.



В тази ситуация се получава така, че данните, които са налични в края на процесорния цикъл са необходими и в неговото начало. За преодоляване на това несъответствие е необходимо инструкцията *beq* да се забави с един процесорен цикъл, чрез внасяне на застой и след това да се използва техниката за пренасочване на данни (*справка – лекцията от предходната седмица*). За целта ще са необходими допълнителни хардуерни схеми, които да могат да откриват и преодоляват посочената ситуация.

2.2. Статично предсказване на преходи

Статичното предсказване на преходи е друг метод за намаляване на застоите при инструкции за преход. В основата му са следните наблюдения:

- Ако се окаже, че прехода няма да се осъществи и следователно програмният брояч няма да се зареди със стойност различна от PC + 4, тогава извлечените инструкции, които се намират непосредствено след инструкцията за преход в паметта, ще продължат да се изпълняват в конвейера, без да се налагат никакви корекции в работата му.
- Ако се окаже, че условието в инструкцията за преход е изпълнено или инструкцията е за безусловен преход, тогава е необходимо извлечените инструкции след инструкцията за преход да се премахнат от конвейера, като се преобразуват в празни инструкции nop (no-operation opcode).

На следващата фигура е представена ситуация при която преходът се осъществява. В четвъртия процесорен цикъл, когато в програмния брояч се зарежда адресът указан в инструкцията за преход beq, е необходимо следващите две инструкции да се преобразуват в инструкции nop.

2.3. Отложени преходи

Третата схема, която е широко използвана в ранните RISC процесори, се нарича „delayed branch“ (отложен преход). При тази схема изпълнението на инструкцията, указана от прехода, (ако прехода се осъществи) или инструкцията която са намира след прехода в паметта (ако прехода не се осъществи) се отлагат с един или два процесорни цикъла в следния ред:

```
инструкция i (за преход)
инструкция 1 (вмъкната за отлагане на прехода) } Времеви отрязък за
инструкция 2 (вмъкната за отлагане на прехода) } отлагане на прехода
указана от прехода инструкция (ако е вярно условието на прехода)
```

или:

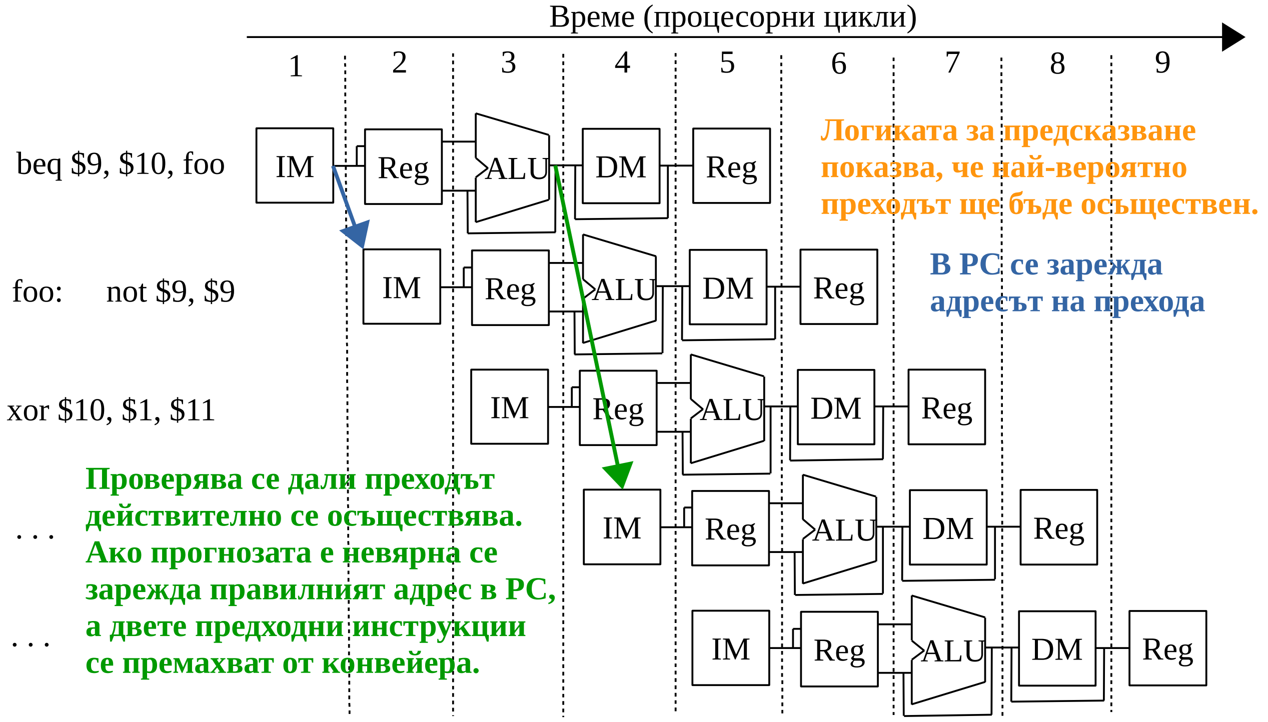
```
инструкция i (за преход)
инструкция 1 (вмъкната за отлагане на прехода) } Времеви отрязък за
инструкция 2 (вмъкната за отлагане на прехода) } отлагане на прехода
инструкция i + 1 (от оригиналната последователност, ако не е вярно условието на прехода)
```

Във времевият отрязък, непосредствено след инструкцията за преход обикновено се вмъкват една или две инструкции, които се изпълняват независимо от това дали прехода ще се осъществи или не. Вмъкнатите инструкции може да бъдат и повече от две, но това е в случай, че преходите внасят по-големи закъснения в конвейера. Кои да бъдат вмъкнатите инструкции, следващи прехода се решава от компилатора. Той трябва да избере валидни и полезни инструкции, които да постави непосредствено след инструкцията за преход. Така, вместо да се внасят застои в процесорните цикли след инструкцията за преход, компилаторът избира независими от прехода инструкции от различни места в кода на програмата, които да вмъкне във времевия отрязък, в който, иначе, би следвало да има застои. Избягвайки застоите ефективността на конвейера се увеличава. Тук сложността се прехвърля върху компилатора, който трябва да се съобразен с работата на конвейера в конкретната процесорна архитектура.

2.4. Динамично предсказване на преходи

В по-новите RISC процесори схемата на отложени преходи не се прилага, защото усъвършенстваната хардуерна технология позволява без затруднения да се реализира динамично предсказване на преходите.

Какъв ще бъде резултатът от проверката на условието и съответно какъв адрес ще трябва да се зареди в програмния брояч, при изпълнение на инструкция за условен преход, може да се предскаже като се използват някои евристични (базирани на опит) техники. За целта е необходимо да се добави памет под формата на буфер или таблица, в която се записва историята – дали прехода е бил осъществен или не при предишните изпълнения на инструкцията за условен преход.



На базата на поведението на прехода при предишните му изпълнения, логиката за динамично предсказване извлича най-вероятната инструкция, която да последва в конвейера инструкцията за условен преход. Резултатите от изпълнението на спекулативно извлечената инструкция не се записват в регистровия файл, докато не се изпълни инструкцията за преход и не се установи, отговаря ли прогнозата на действителния резултат от проверката на условието в инструкцията за преход. Предсказването на преходите не може да бъде със 100% точност, така че ефективността на конвейера зависи от дела на точните предсказвания.

3. Стратегии за предсказване на преходи

Сега, ще разгледаме някои стратегии, които се прилагат при предсказване поведението на преходите. Прогнозирането е в две посоки: предсказване на резултата от условието на прехода и предсказване на целевия адрес, които следва да се зареди в програмния брояч, ако прехода се осъществи. Различните стратегии имат различна точност. По-сложните и усъвършенствани стратегии са с по-голяма точност. Стратегиите с по-голяма точност водят до по-ефективно изпълнение на инструкциите в конвейера и оттам до повишаване на неговата производителност.

3.1. Предсказване на резултата от условието на прехода

Задачата тук е да се предскаже дали преходът ще бъде осъществен или не.

3.1.1. Статично предсказване на преходите. Статично предсказване означава, че прогнозата ще бъде една и съща за всяка инструкция за преход, влязла в конвейера. Има три стратегии за статично прогнозиране на преходи:

1. **Преходът никога не се осъществява** – Приема се, че всеки преход няма да бъде осъществен. Така изпълнението на инструкциите в конвейера продължава все едно, че преходът не е изпълнен. Тук, обаче, трябва да се следи да не би някоя инструкция да промени състоянието на процесора преди окончателно да е станало ясно как ще продължи изпълнението на програмата след прехода. Оттук и необходимостта от допълнителна хардуерна логика, която да следи дали състоянието на процесора е променено с инструкция и как да се „откаже” такава промяна.

   В класическия конвейер с пет етапа тази стратегия с предварително приемане, че прехода е неосъществен се реализира, като извличането на инструкции продължава, все едно, че не се изпълнява инструкция за преход и не се случва нищо необичайно. Ако, обаче проверката на условието в прехода успее, то извлечените инструкции не трябва да продължават, затова те се заменят с „празни“ инструкции и извличането от целевия адрес се рестартира.
2. **Преходът винаги се осъществява** – По-ефективна и малко по-сложна е стратегията, при която всеки преход се третира като осъществен. Веднага след като инструкция за преход се декодира, се приема, че условието в прехода е вярно и започва извличане и изпълнение на инструкцията, намираща се на указания от прехода адрес. Ако проверката на условието в прехода действително успее, то забавянето ще се намали с един или два процесорни цикъла, в зависимост от това дали условието в прехода и целевият адрес се изчисляват в края на етапа декодиране (ID), или в края на етапа изпълнение (EXE).

И в двете стратегии с предварително приемане – преходът е неосъществен или преходът е осъществен – точността е ниска, но компилаторът може да подобри производителността, като организира кода, така че най-често да се среща случаят, съответстващ на избраната хардуерна схема.

3. **Евристично (базирано на опит) предсказване на преходите.** Тази стратегия е по-точна. При евристичното предсказване трябва да има натрупани данни за поведението на преходите. Данните се събират, като множество програми се стартират многократно. От тези данни се прогнозира каква е статистическата вероятност даден преход да се осъществи или не. Оказва се например, че преходите назад в кода на програмата с голяма степен на вероятност се

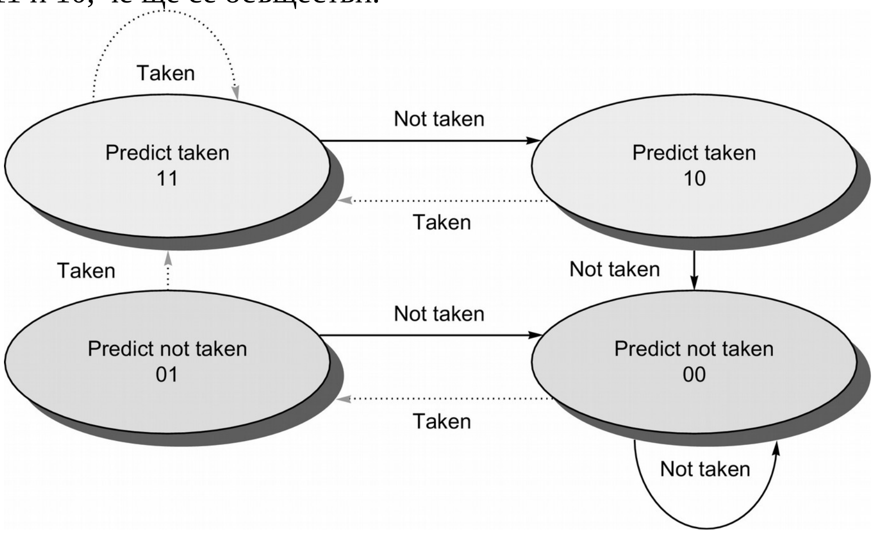
осъществяват, тъй като това най-често са програмни цикли, а вероятността да се осъществят преходите напред в кода е малка. Това може да се използва при изпълнението на преходите в конвейера.

3.1.2. Динамично предсказване на преходи и буфери за предсказване на преходи

При извличане на инструкция за преход, преди за започне нейното изпълнение, хардуерна схема дава прогноза за това дали преходът ще се изпълни или не. Динамичното прогнозиране постига по-висока точност спрямо статичното.

Бимодално предсказване. Най-простата техника за динамично предсказване е буфера за предсказване на преходи или таблица с история на преходите. Буферът за предсказване на преходи е малка памет индексирана чрез младшата част от адреса на инструкцията за преход. Паметта съдържа бит, който показва дали при последното изпълнение, преходът е осъществен или не. Това е най-простият вид буфер и точността му не е голяма.

Предсказване с хистерезис. По-често се използва буфер с два бита. В този случай, прогнозата дали преходът ще се осъществи или не се променя едва, ако два поредни пъти текущата прогноза се окаже невярна. Фигура по-долу показва промяната на битовете при осъществяване или не на преход. При 00 или 01 прогнозата е, че прехода няма да се осъществи, а при 11 и 10, че ще се осъществи.

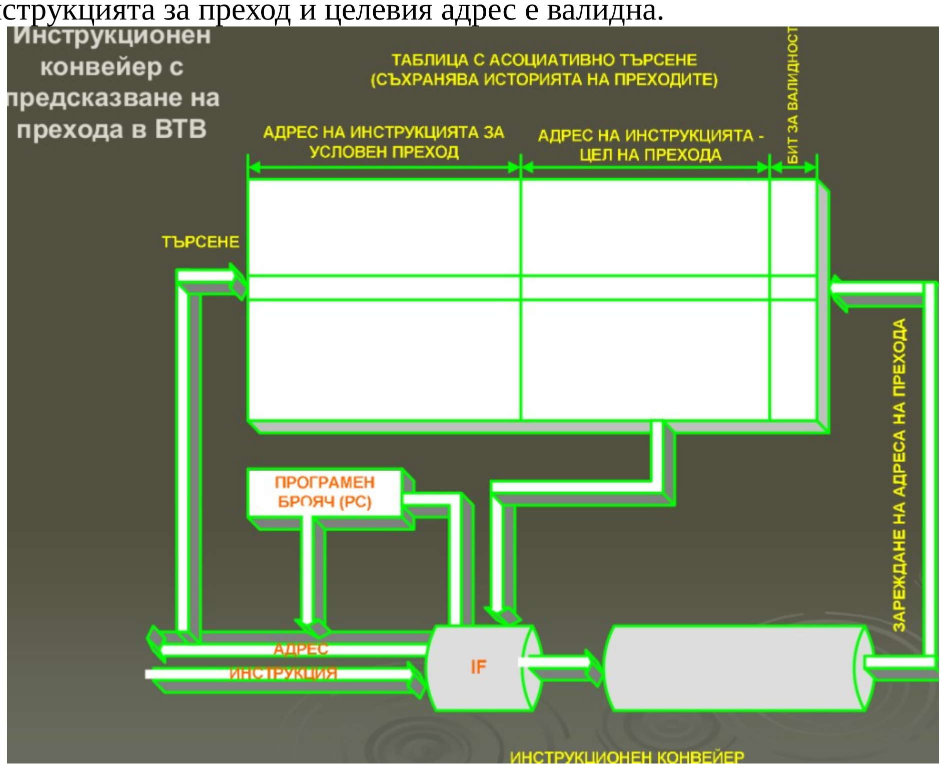


Чрез използване на два, вместо на един бит, преходите, който предимно се осъществяват или тези които предимно не се осъществяват, (а това са голямата част от преходите), по-рядко ще бъдат предвидени неправилно, отколкото при предсказване с 1-битов буфер. Двата бита се използват за кодиране на четири състояния в системата. Двубитовата стратегия е частен случай на по-обща схема, която има n-битов брояч за всеки запис в буфера за прогнозиране. При n-бита, броячът може да брои от 0 до 2^{n-1}: когато стройността в брояча е по-голяма или равена на половината от неговата максимална стойност (2^{n-1}), преходът се прогнозира като осъществен; в противен случай се прогнозира като неосъществен. Изследванията показват, че двубитовата схема е с почти същата точност както n-битовата, поради което в повечето конвейери се разчита на схеми с 2-битови броячи.

3.2. Предсказване на целевия адрес на прехода

Предсказването на целевия адрес може а бъде толкова важно, колкото и предсказването на това дали преходът ще бъде осъществен или не. При безусловен преход или осъществен условен преход, или във всеки друг случай, когато се променя последователността на инструкциите записани в паметта, адресът на следващата инструкция трябва да бъде възможно най-скоро на разположение в етапа IF. Основната техника за предсказване на целевият адрес се нарича Branch Target Buffer (BTB).

BTB представлява таблица, която съдържа: от една страна, адресите на инструкциите за условен преход, а от друга – адресите на инструкциите, които трябва да се изпълнят ако условният преход се осъществи. Когато един преход се осъществи, изчисленият целеви адрес на прехода се записва в таблицата заедно с адреса на инструкцията за преход от програмния брояч (PC). Когато е необходимо да се направи предсказване на целевия адрес на прехода, той се взема от BTB. Забележете, че BTB съхранява само целевите адреси на осъществени преходи; при не осъществяване на прехода изпълнението продължава със следващата инструкция. Таблицата съхранява бит за валидност, за да се гарантира, че асоциацията между адреса на инструкцията за преход и целевия адрес е валидна.



Конвейерното изпълнение на инструкции, проблемите, които възникват при него и тяхното преодоляване е в основата на т. нар. паралелизъм на ниво инструкции. В следващата лекция предстои да се запознаем с още техники и алгоритми, които усъвършенстват паралелното, застъпено изпълнение на инструкциите в една програма, с цел постигане на възможно най-голяма производителност.