КАТЕДРА: КОМПЮТЪРНИ СИСТЕМИ И ТЕХНОЛОГИИ
ДИСЦИПЛИНА: КОМПЮТЪРНИ АРХИТЕКТУРИ

ЛАБОРАТОРНО УПРАЖНЕНИЕ № 6

ТЕМА: КОНВЕЙЕРНО ИЗПЪЛНЕНИЕ НА ИНСТРУКЦИИТЕ В ПРОЦЕСОРА – II ЧАСТ

Цел: Да даде по-задълбочени практически знания на студентите за възникването и избягването на състезанията между инструкциите при тяхното конвейерно изпълнение в процесора.

Задачи за изпълнение: Конвейерното изпълнение на инструкциите може да ускори работата на процесорите до N пъти, спрямо неконвейерното, където N е броят на етапите в конвейера. Състезанията между инструкциите в конвейера (*виж лекцията от тази седмица*), ограничават неговата ефективност и на практика не се постига максимално възможното ускорение. В предложените по-долу задачи ще изследваме по-детайлно проблемите при конвейерно изпълнение в процесора.

Задача 1: Програмите в предходното упражнение включваха по една инструкция, а задачата бе да се проследи тяхното изпълнение в конвейера. Реална полза от използването на конвейер в процесора има тогава, когато застъпено се изпълняват няколко инструкции в него, така че хардуерът да е зает да изпълнява няколко инструкции едновременно.

Сега ще направим експеримент с малка програма с няколко инструкции и ще изследваме как на различните етапи в конвейера инструкциите се изпълняват паралелно.

Разгледайте следния код:

```
.data
           x: .word 0x45
.text
Start:     la $9, x          # Зарежда адреса на x
           nop               # Инструкция без изпълнение на някаква операция
           nop               # Инструкция без изпълнение на някаква операция
           nop               # Инструкция без изпълнение на някаква операция
Repeat:    lw $8, 0x0($9)    # Чете от паметта
           nop               # Инструкция без изпълнение на някаква операция
           nop               # Инструкция без изпълнение на някаква операция
           nop               # Инструкция без изпълнение на някаква операция
           sw $8, 0x0($9)    # Запис в паметта
           b Repeat          # Повтаря четеното и записа
           nop               # Инструкция без изпълнение на някаква операция
           nop               # Инструкция без изпълнение на някаква операция
           li $8, 0          # Нулира регистъра
end:                         # Маркира края на програмата
```

Компилирайте и заредете програмата в симулатора *MipsPipeS.exe*, както бе описано в предходното упражнение. Изпълнете я стъпка по стъпка. Проследете кога започва изпълнението на инструкциите и кога се изчисляват и записват резултатите. Също така отбележете колко инструкции се изпълняват едновременно в различните етапи на конвейера.

Задача 2: Променете следния код, като присвоите различни стойности на *$8* и *$9*. Компилирайте програмата и я изпълнете постъпково в симулатора *MipsPipeS.exe*.

```
.text
start:
      add $10, $8, $9
      add $12, $10, $11
      nop
      nop
      nop
.end start
```

Забележка: Следвайте инструкциите в предходното упражнение.

- След колко процесорни цикъла ще се запише стойността на резултата от първата инструкция за събиране в целевия регистър *$10*?

- След колко процесорни цикъла ще е необходима стойността от *$10* във втората инструкция?
- Каква зависимост възниква? Как се нарича този вид зависимости?

Задача 3: Зависимостта между инструкциите, представена в задача 2 може да бъде преодоляна чрез:

1. пренареждане на кода;
2. вмъкване на празни инструкции NOP;
3. „pipeline interlock“ (открива на зависимостите и „заключва“ конвейера);
4. пренасочване.

Обяснете кога могат да се използват първите три метода и как работят.

За преодоляване на зависимостите по данни, в конвейера на MIPS процесорите, са добавени схеми за пренасочване на данните. В софтуера за симулация, който използваме също има внедрено пренасочване. Досега използвахме варианта на програмата за симулация, в която няма пренасочване – *MipsPipeS.exe*. Сега ще използваме програмата *MipsPipeXL.exe*, намираща се в директория *C:\MipsIT\bin*, в която има реализирано пренасочване. Изпълнете постъпково програмата от задача 2, като вместо *MipsPipeS.exe* използвате *MipsPipeXL.exe* и проследете как работи пренасочването.

Отговорете на следните въпроси:

- Кога (при какви условия) схемата за управление на пренасочването (Forwarding Unit) задейства мултиплексорите (Mux), намиращи се преди аритметично-логическото устройство (ALU), за да превключват и пренасочват данните?
- За пренасочените данни от примера, определете: от къде се прочитат първоначално и къде се записват накрая?

Задача 4: Променете следващата програмата, така че да присвоите различни стойности на *$8* и *$9*. Регистърът *$10*, трябва да съдържа адрес на клетка от паметта, в която предварително сте записали стойност (*вижте „Индиректно и индексно адресиране“ от упражнение 4*). Компилирайте и направете симулация в *MipsPipeS.exe* като изпълните програмата постъпково.

```
.text
start:
      lw  $8, 0($10)
      add $9, $9, $8
      nop
      nop
      nop
.end start
```

- След колко процесорни цикъла от началото на изпълнението на инструкцията *lw* в регистъра *$8*, ще се зареди указаната стойност?
- След колко процесорни цикъла е необходима стойността на *$8*, в инструкцията *add*?
- Каква е зависимостта в този случай? Как се нарича?
- Какви са вариантите за преодоляване на този вид зависимости?
- Като изпълните програмата постъпково в *MipsPipeXL.exe*, проверете как тези зависимости се преодоляват в конвейера с пренасочване?

Задача 5: Променете следващата програма, като присвоите еднакви стойности на *$8* и *$9*.

```
.text
start:
      nop
      nop
      beq  $8, $9, start
      addi $8, $8, 1
      nop
      nop
      nop
.end start
```

- Компилирайте кода и симулирайте постъпково в програмата за симулация на конвейер без пренасочване (*MipsPipeS.exe*).
- Колко процесорни цикъла отнема инструкцията за преход, преди да се осъществи прехода? Какво се случва със следващата инструкция *addi*, докато инструкцията за преход се изпълнява?
- Каква е зависимостта в този случай? Как се нарича?
- Посочете как може да се преодолее възникналата в програмата зависимост?
- Симулирайте постъпково в конвейера с пренасочване (*MipsPipeXL.exe*). Преодолява ли се зависимостта, породена от инструкцията *beq*?