Политики на подаване на инструкциите при суперскаларни процесори

(ЛУ 12)

Терминът **подаване на инструкция (instruction issue)** обуславя процесът на инициализиране на изпълнение на дадена инструкция във функционалните устройства на процесора(ФУ).

Терминът **политика на подаване на инструкциите (instruction issue policy)** характеризира начинът по който се подават инструкции.

При **политиките на подаване на инструкциите** се открояват 3 важни момента:

1) Редът в който инструкциите биват извличани (fetch)
2) Редът в който инструкциите биват изпълнени (execute)
3) Редът в който инструкциите обновяват клетките от паметта/кеша които заемат (completion)

С оглед на гореспоменатите моменти можем да категоризираме **политиките на подаване** в 3 категории:

1) In-order issue, in-order completion
2) In-order issue, out-of-order(ooo) completion
3) Out-of-order issue, out-of-order completion

In-order issue, in-order completion - най-лесната за реализация от 3-те, тук инструкциите се подават в ред в който биват извлечени и резултатът от изпълнението им се запазва в същия ред.

In-order issue, out-of-order completion - тук подаването е под ред, докато записването може да бъде в разбъркан ред. Постига се пълна натовареност на **ФУ** поради "свободата" на записване.

Out-of-order issue, out-of-order completion - Постига се със буфер наречен **инструкционен прозорец (instruction window)**, в който се запазват декодирани инструкции готови да бъдат подадени при предоставена възможност. Те могат да бъдат подавани в разбъркан ред.

Важно: Преди да се подаде инструкция, трябва да е сигурно че тя няма да предизвика зависимости, било то **функционални** (споделяне на едно ѝ също **ФУ**), или **даннови** (промяната на данните от една инструкция би предизвикала некоректно изпълнение на последващи такива и нарушаване на програмната последователност).

На фигурата по-долу са показани изпълненията на 3-те категории.

Decode		Execute			Write		Cycle
I1	I2						1
I3	I4	I1	I2				2
I3	I4	I1					3
	I4			I3	I1	I2	4
I5	I6			I4			5
	I6		I5		I3	I4	6
			I6				7
					I5	I6	8

(a) In-order issue and in-order completion

Decode		Execute			Write		Cycle
I1	I2						1
I3	I4	I1	I2				2
	I4	I1		I3	I2		3
I5	I6			I4	I1	I3	4
	I6		I5		I4		5
			I6		I5		6
					I6		7

(b) In-order issue and out-of-order completion

Decode		Window	Execute			Write		Cycle
I1	I2							1
I3	I4	*I1, I2*	I1	I2				2
I5	I6	*I3, I4*	I1		I3	I2		3
		I4, I5, I6		I6	I4	I1	I3	4
		I5		I5		I4	I6	5
						I5		6

(c) Out-of-order issue and out-of-order completion

Задача 1:

16.3 Consider the following assembly language program:

```
I1: Move R3, R7          /R3 ← (R7)/
I2: Load R8, (R3)        /R8 ← Memory (R3)/
I3: Add R3, R3, 4        /R3 ← (R3) + 4/
I4: Load R9, (R3)        /R9 ← Memory (R3)/
I5: BLE R8. R9. L3       /Branch if (R9) > (R8)/
```

This program includes WAW, RAW, and WAR dependencies. Show these.

Задача 2:

Decode		Execute			Write		Cycle
I1	I2						1
	I2			I1			2
	I2			I1			3
I3	I4		I2				4
I5	I6		I4	I3	I1	I2	5
I5	I6			I3			6
		I5	I6		I3	I4	7
		I5					8
					I5	I6	9

Figure 16.14 An In-Order Issue, In-Order-Completion Execution Sequence

16.5 Consider the "in-order-issue/in-order-completion" execution sequence shown in Figure 16.14.

a. Identify the most likely reason why I2 could not enter the execute stage until the fourth cycle. Will "in-order issue/out-of-order completion" or "out-of-order issue/out-of-order completion" fix this? If so, which?

b. Identify the reason why I6 could not enter the write stage until the nineth cycle. Will "in-order issue/out-of-order completion" or "out-of-order issue/out-of-order completion" fix this? If so, which?

Задача 3:

16.6 Figure 16.15 shows an example of a superscalar processor organization. The processor can issue two instructions per cycle if there is no resource conflict and no data dependence problem. There are essentially two pipelines, with four processing stages (fetch, decode, execute, and store). Each pipeline has its own fetch decode and store unit. Four functional units (multiplier, adder, logic unit, and load unit) are available for use in the execute stage and are shared by the two pipelines on a dynamic basis. The two store units can be dynamically used by the two pipelines, depending on availability at a particular cycle. There is a lookahead window with its own fetch and decoding logic. This window is used for instruction lookahead for out-of-order instruction issue.

Consider the following program to be executed on this processor:

```
I1: Load R1, A      /R1 ← Memory (A)/
I2: Add R2, R1      /R2 ← (R2) + R(1)/
I3: Add R3, R4      /R3 ← (R3) + R(4)/
I4: Mul R4, R5      /R4 ← (R4) + R(5)/
I5: Comp R6         /R6 ← (R6)/
I6: Mul R6, R7      /R6 ← (R6) × R(7)/
```

a. What dependencies exist in the program?

b. Show the pipeline activity for this program on the processor of Figure 16.15 using in-order issue with in-order completion

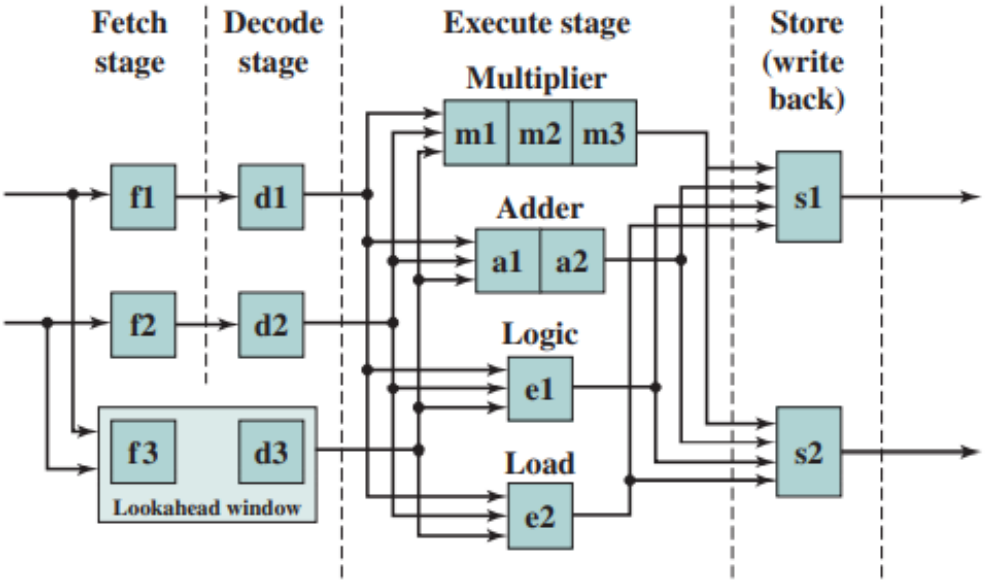

Figure 16.15 A Dual-Pipeline Superscalar Processor

c. Repeat for in-order issue with out-of-order completion.
d. Repeat for out-of-order issue with out-of-order completion.

Приемаме че: Има реализиран forwarding, Няма register renaming, След декодиране инструкциите се прехвърлят в **instruction window** в същия такт. Размерът на instruction window е безкрайно голям.