Excepções

Excepções

Uma excepção assinala uma circunstância excepcional, ocorrida durante o processamento, que requer atenção com urgência

Uma excepção pode ter origem interna ou externa ao processador

As excepções externas ao processador são também conhecidas como interrupções (*interrupts*)

Exemplos

Evento causador	Origem	Tipo
Overflow aritmético	Interna	Excepção
Uso de uma instrução inválida	Interna	Excepção
Chamada ao sistema operativo	Interna	Excepção
Pedido de um dispositivo de I/O	Externa	Interrupção
Relógio (preempção no SO)	Externa	Interrupção
Problema de *hardware*	Interna ou externa	Excepção ou interrupção

Tratamento de excepções

O tratamento (ou atendimento) de excepções consiste em

1. Interromper a execução do programa corrente

   É necessário limpar o *pipeline* e guardar o endereço da primeira instrução cuja execução não foi completada

2. Executar o código do SO que lida com a excepção ocorrida (ou levantada, ou gerada)

   O processador vai executar as instruções localizadas a partir de um endereço pré-determinado fixo ou de um endereço que depende da excepção em causa (neste caso designam-se por *vectored interrupts*)

3. Retomar a execução do programa ou abortá-la

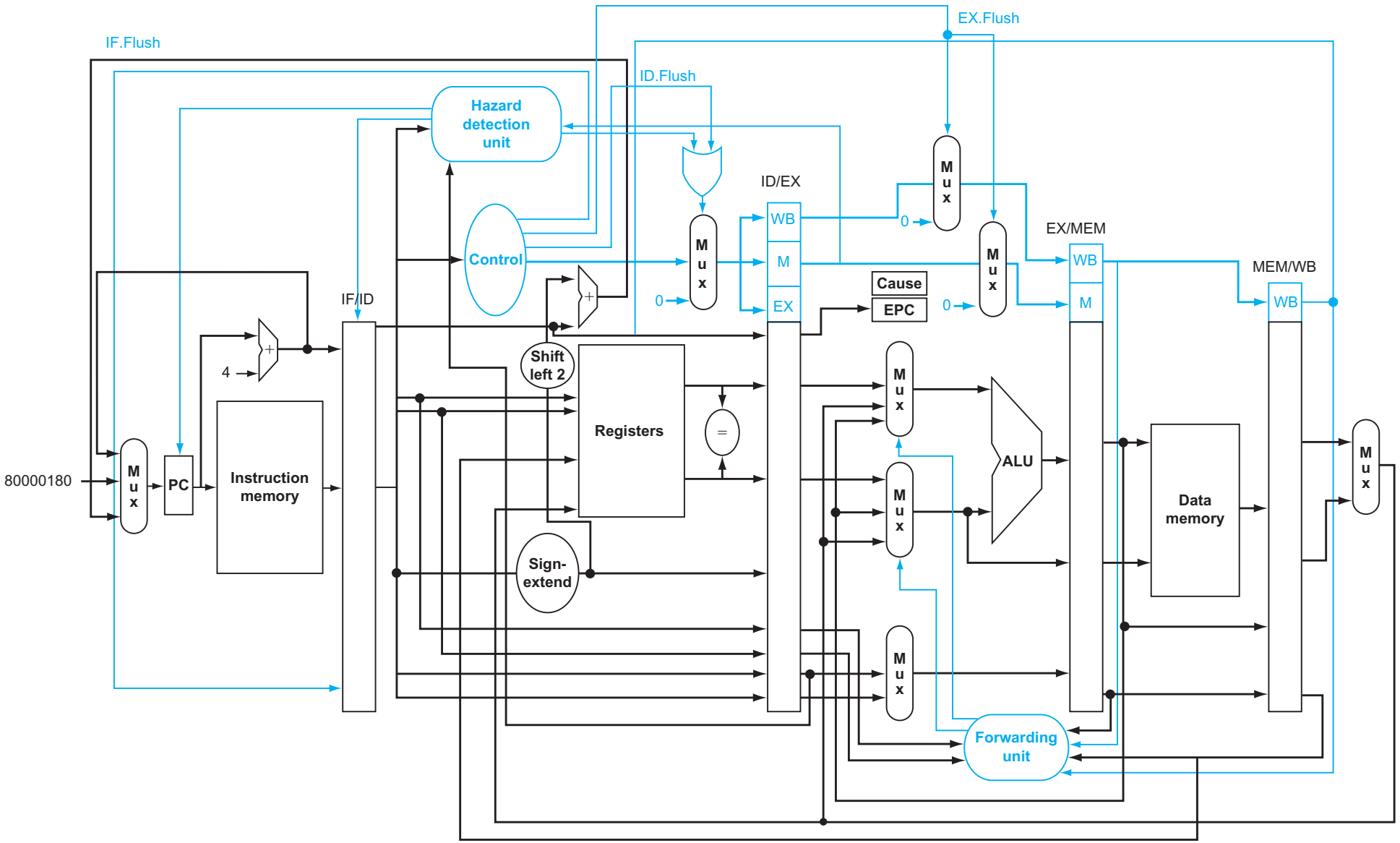
   Se a excepção se deveu a um erro do programa, ele é abortado

Tratamento de excepções no MIPS

Para tratamento de excepções, o *pipeline* MIPS tem

- Um registo EPC (*exception* PC) onde é guardado o endereço da instrução que esteve na origem da excepção
  (esse endereço + 4, na realidade)
- Um registo Cause onde é guardada a causa da excepção (*overflow* aritmético, instrução inválida, . . .)
- Os sinais IF.Flush, ID.Flush e EX.Flush para limpar os registos IF/ID, ID/EX e EX/MEM, respectivamente, do *pipeline* (ou somente os sinais de controlo nos registos)
- O endereço reservado 8000 0180_{16}, onde começa o código que trata a generalidade das excepções

Pipeline com tratamento de excepções



Tratamento de excepções no *pipeline* MIPS

Um exemplo

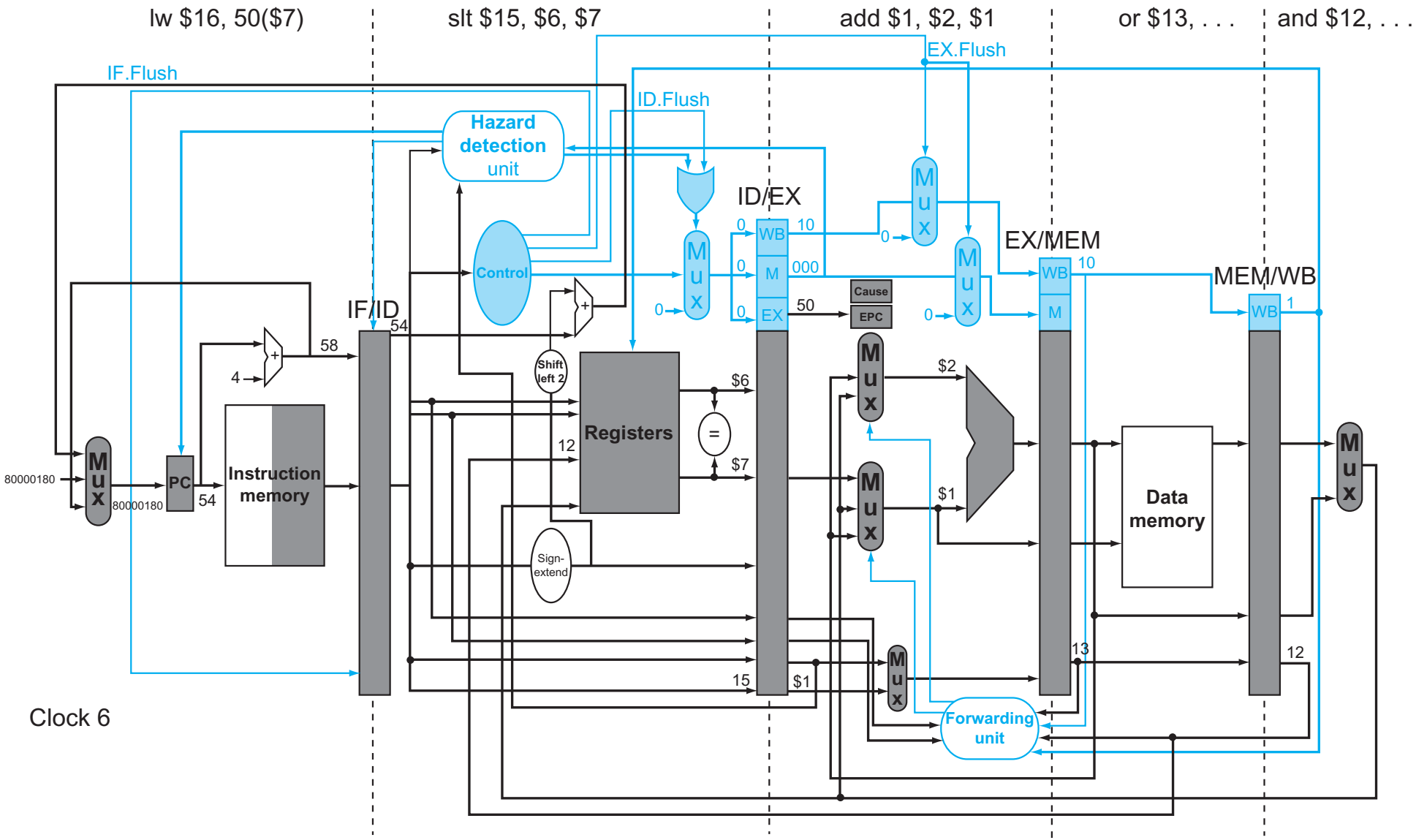
```
40_16   sub $11, $2, $4
44_16   and $12, $2, $5
48_16   or  $13, $2, $6
4C_16   add  $1, $2, $1   ← overflow
50_16   slt $15, $6, $7
54_16   lw  $16, 50($7)
...

8000 0180_16   sw  $26, 1000($0)
8000 0184_16   sw  $27, 1004($0)
...
```

(Os registos $26 ($k0) e $27 ($k1) são reservados para uso pelo sistema operativo)

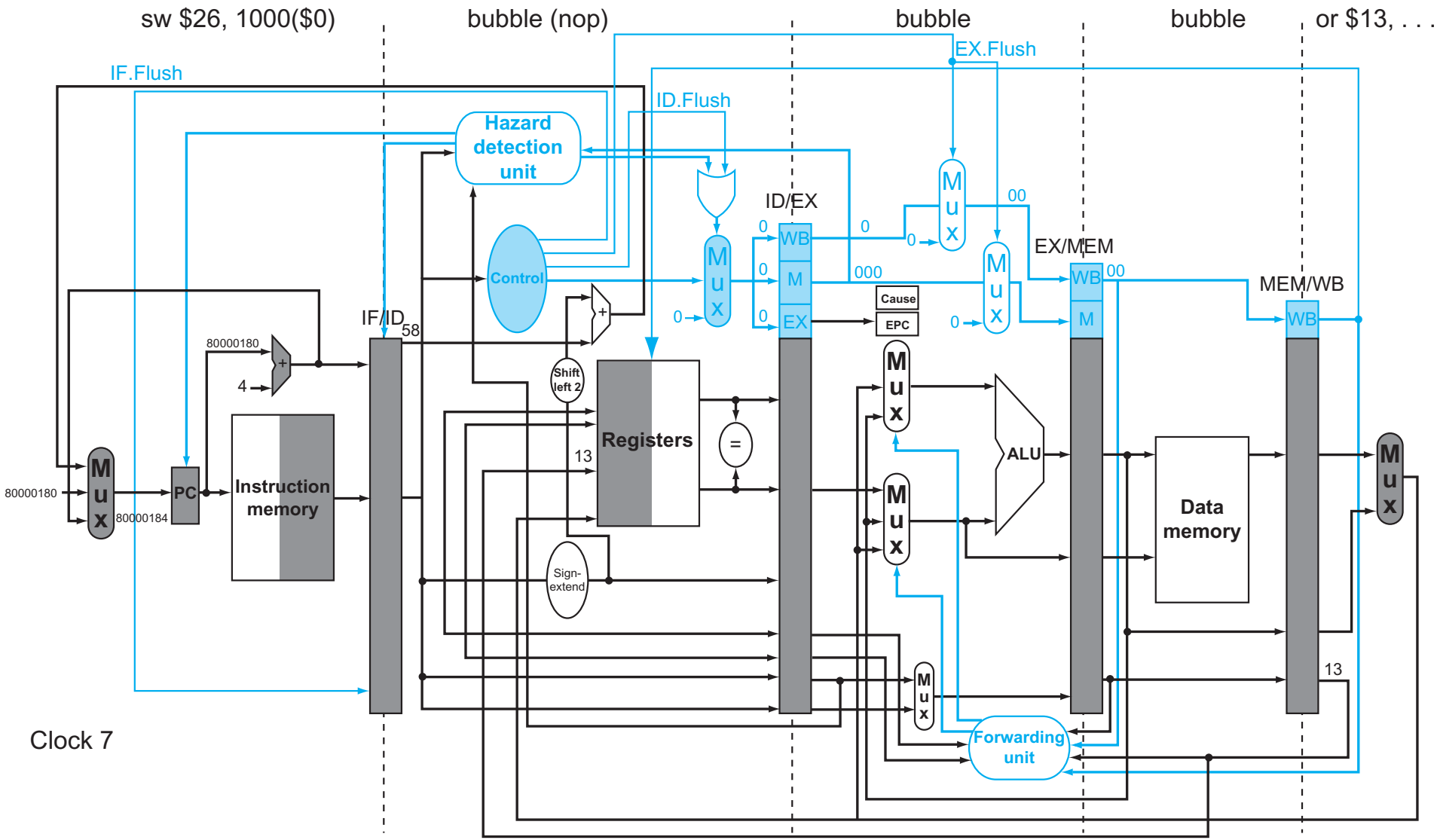
Tratamento de excepções no *pipeline* MIPS

Estado do *pipeline* no ciclo de relógio em que é gerada a excepção



Tratamento de excepções no *pipeline* MIPS

O ciclo seguinte



Excepções simultâneas

Tratamento de excepções simultâneas

Quando no mesmo ciclo de relógio ocorrem múltiplas excepções, elas são atendidas começando pela que corresponde à instrução mais avançada no *pipeline*

Instruction-level parallelism

(Paralelismo na execução de instruções)

Instruction-level parallelism (ILP)

Paralelismo ao nível (da execução) das instruções

Trata-se do uso de paralelismo na execução das instruções de um programa sequencial num único CPU

Não se trata da execução de programas paralelos, compostos por várias partes que podem ser executadas em paralelo em múltiplos CPUs (ou *cores*)

Formas de *instruction-level parallelism*

Pipelining

- É uma forma de ILP
- Várias instruções estão a ser executadas em cada ciclo de relógio

  Quanto mais profundo é o *pipeline*, maior é o ILP
- Cada instrução está numa fase diferente da execução

Multiple issue

- É outra forma de ILP
- Várias instruções começam a ser executadas em cada ciclo de relógio
- Instruções recorrem a unidades funcionais duplicadas
- O CPI pode tornar-se inferior a 1 (como alternativa, usa-se o IPC, que é o número médio de instruções executadas por ciclo)

Multiple issue

Várias instruções podem começar a ser executadas (ou podem ser lançadas) em cada ciclo de relógio

A *issue width* do processador é o número de instruções que podem ser lançadas em simultâneo

As instruções candidatas a lançamento simultâneo encontram-se nos *issue slots* do processador

As instruções que são efectivamente lançadas em simultâneo constituem um *issue packet*

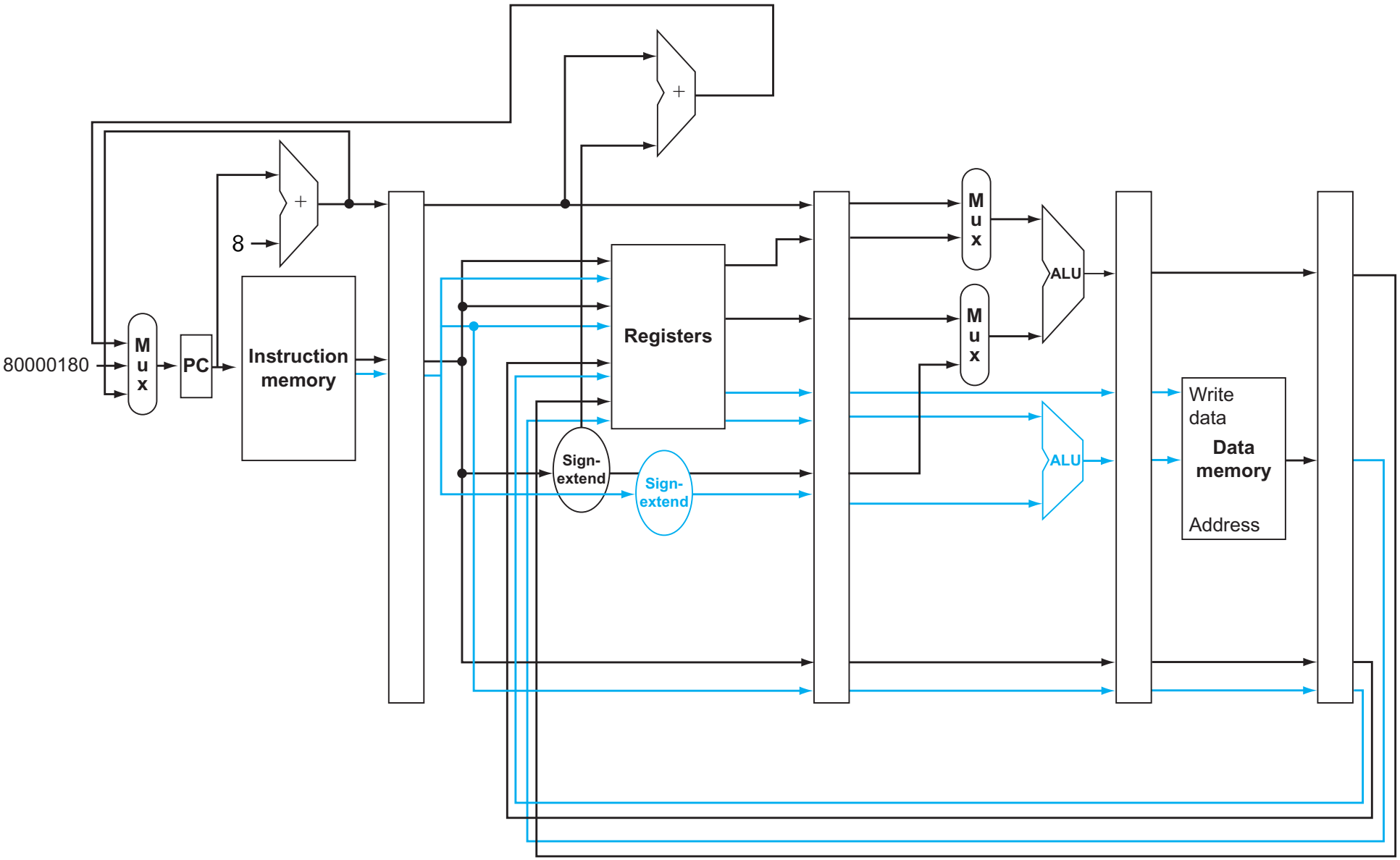
Multiple issue estático

É o compilador que decide que instruções serão executadas em simultâneo, organizando-as nos *issue slots* do processador

Pode caber ao compilador reduzir ou garantir que não existirão conflitos (de dados, de controlo ou estruturais)

As instruções nos *issue slots* (que vão constituir o *issue packet*) podem ser encaradas como uma única grande instrução, apelidada de *Very Long Instruction Word* (VLIW)

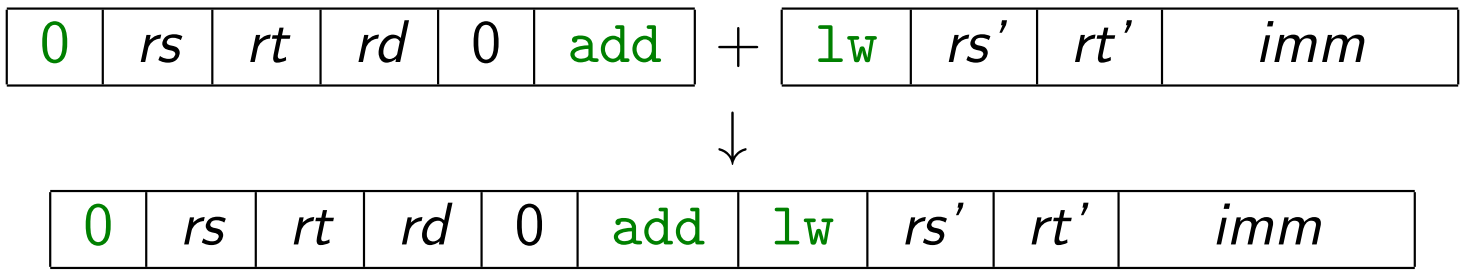
Pipeline MIPS com *2-way multiple issue*



MIPS com VLIW

Processador MIPS com *2-way multiple issue*, ou *double issue*

Pode executar uma instrução de acesso à memória em simultâneo com uma instrução aritmética ou um salto condicional



Funcionamento do *pipeline* com *double issue* estático

Instruction type	Pipe stages							
ALU or branch instruction	IF	ID	EX	MEM	WB			
Load or store instruction	IF	ID	EX	MEM	WB			
ALU or branch instruction		IF	ID	EX	MEM	WB		
Load or store instruction		IF	ID	EX	MEM	WB		
ALU or branch instruction			IF	ID	EX	MEM	WB	
Load or store instruction			IF	ID	EX	MEM	WB	
ALU or branch instruction				IF	ID	EX	MEM	WB
Load or store instruction				IF	ID	EX	MEM	WB

Código para o MIPS com VLIW

Código original

```
Loop:   lw   $t0, 0($s1)        # $t0 ← elemento do vector
        addu $t0, $t0, $s2      # soma o escalar em $s2
        sw   $t0, 0($s1)        # guarda o resultado
        addi $s1, $s1, -4       # decrementa endereço
        bne  $s1, $zero, Loop   # repete se $s1 ≠ 0
```

Código reorganizado para *2-way multiple issue* estático

	ALU or branch instruction	Data transfer instruction	Clock cycle
`Loop:`		`lw $t0, 0($s1)`	1
	`addi $s1,$s1,-4`		2
	`addu $t0,$t0,$s2`		3
	`bne $s1,$zero,Loop`	`sw $t0, 4($s1)`	4

$$\text{CPI} = \frac{4}{5} = 0.8 \text{ (mínimo 0.5)} \qquad \text{IPC} = \frac{5}{4} = 1.25 \text{ (máximo 2)}$$

Loop unrolling

Código do ciclo desdobrado (ou desenrolado) quatro vezes e reorganizado para *2-way multiple issue*

	ALU or branch instruction		Data transfer instruction		Clock cycle
`Loop:`	`addi`	`$s1,$s1,-16`	`lw`	`$t0, 0($s1)`	1
			`lw`	`$t1,12($s1)`	2
	`addu`	`$t0,$t0,$s2`	`lw`	`$t2, 8($s1)`	3
	`addu`	`$t1,$t1,$s2`	`lw`	`$t3, 4($s1)`	4
	`addu`	`$t2,$t2,$s2`	`sw`	`$t0, 16($s1)`	5
	`addu`	`$t3,$t3,$s2`	`sw`	`$t1,12($s1)`	6
			`sw`	`$t2, 8($s1)`	7
	`bne`	`$s1,$zero,Loop`	`sw`	`$t3, 4($s1)`	8

Registos renomeados para evitar falsas dependências (*name dependence* ou *antidependence*)

$$\text{CPI} = \frac{\text{nº ciclos}}{\text{nº instruções}} = \frac{8}{14} = 0.57 \qquad \text{IPC} = \frac{\text{nº instruções}}{\text{nº ciclos}} = \frac{14}{8} = 1.75$$

Multiple issue dinâmico

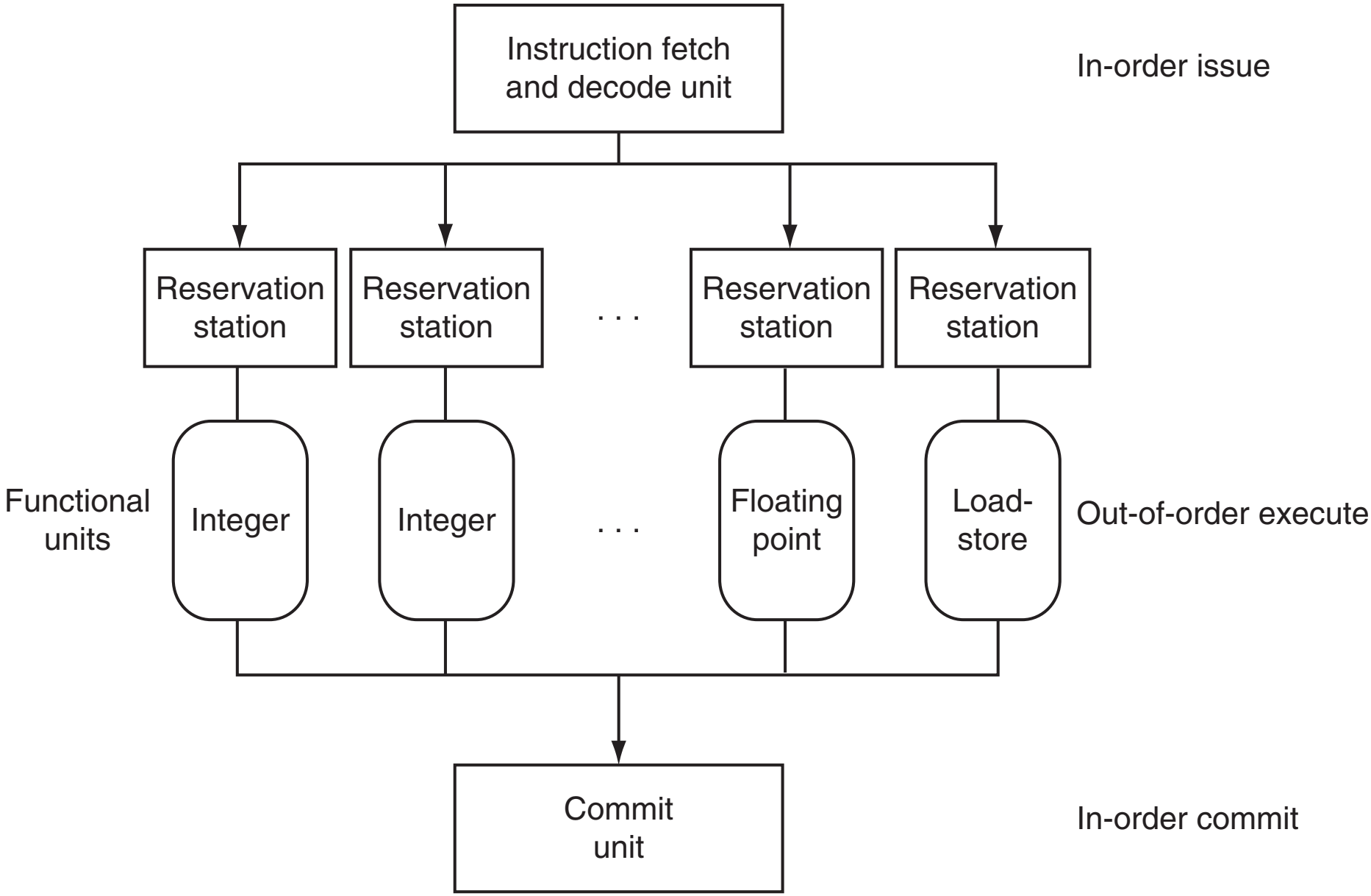
Processador analisa as instruções nos *issue slots* e decide quantas lançará em simultâneo

É o processador que lida com os conflitos estruturais, de dados e de controlo, garantindo a correcção da execução

Um processador superescalar é um processador com *multiple issue* dinâmico

Nalguns casos, as instruções podem ser executadas fora de ordem (*dynamic pipeline scheduling*)

Implementação de execução fora de ordem



Execução especulativa de instruções

Execução especulativa, ou especulação, consiste em decidir que instruções executar assumindo que uma instrução anterior terá um determinado efeito, por exemplo

- que um salto condicional será (ou não) efectuado
- que um *store* não acederá ao mesmo endereço que um *load* que o segue

A execução especulativa permite aumentar o ILP

Quando a especulação tem origem no compilador, este inclui código para verificar que o resultado foi o esperado

Evolução das características dos processadores

Microprocessor	Year	Clock Rate	Pipeline Stages	Issue Width	Out-of-Order/ Speculation	Cores/ Chip	Power	
Intel 486	1989	25 MHz	5	1	No	1	5	W
Intel Pentium	1993	66 MHz	5	2	No	1	10	W
Intel Pentium Pro	1997	200 MHz	10	3	Yes	1	29	W
Intel Pentium 4 Willamette	2001	2000 MHz	22	3	Yes	1	75	W
Intel Pentium 4 Prescott	2004	3600 MHz	31	3	Yes	1	103	W
Intel Core	2006	2930 MHz	14	4	Yes	2	75	W
Intel Core i5 Nehalem	2010	3300 MHz	14	4	Yes	2	87	W
Intel Core i5 Ivy Bridge	2012	3400 MHz	14	4	Yes	8	77	W

AMD Opteron X4 (Barcelona)

Instruções x86 são traduzidas para *RISC operations* (Rops)
(a Intel chama-lhes *micro-operations*)

Processador superescalar com especulação, lança até 3 Rops por ciclo

Pode ter até 106 Rops em execução

Implementa os 16 registos da arquitectura x86-64 através de 72 registos físicos

Pipeline do X4