处理器体系结构

第四章 处理器的微架构D

--中断与异常的处理

(Micro-processor Architecture)

4.4 中断与异常

一些处理器架构不对“中断、异常”作区分，统一称作中断，比如x86

MIPS架构中，中断与异常区别：

- 中断来自外部I/O
- 异常来自CPU内部，比如未定义的指令、溢出、系统调用

Type of event	From where?	MIPS terminology
I/O device request	External	Interrupt
Invoke the operating system from user program	Internal	Exception
Arithmetic overflow	Internal	Exception
Using an undefined instruction	Internal	Exception
Hardware malfunctions	Either	Exception or interrupt

中断与异常的共性：

- 都会改变控制流
- 都会导致性能降低

4.4 MIPS中的中断/异常

在MIPS中，异常的管理是由系统控制协处理器(CP0)完成的

EPCWrite控制信号

- **设置EPC(Exception Program Counter)，标记产生异常的位置**
- **用“原因寄存器(Cause Register, 32bits)”纪录异常产生的原因**

CauseWrite控制信号

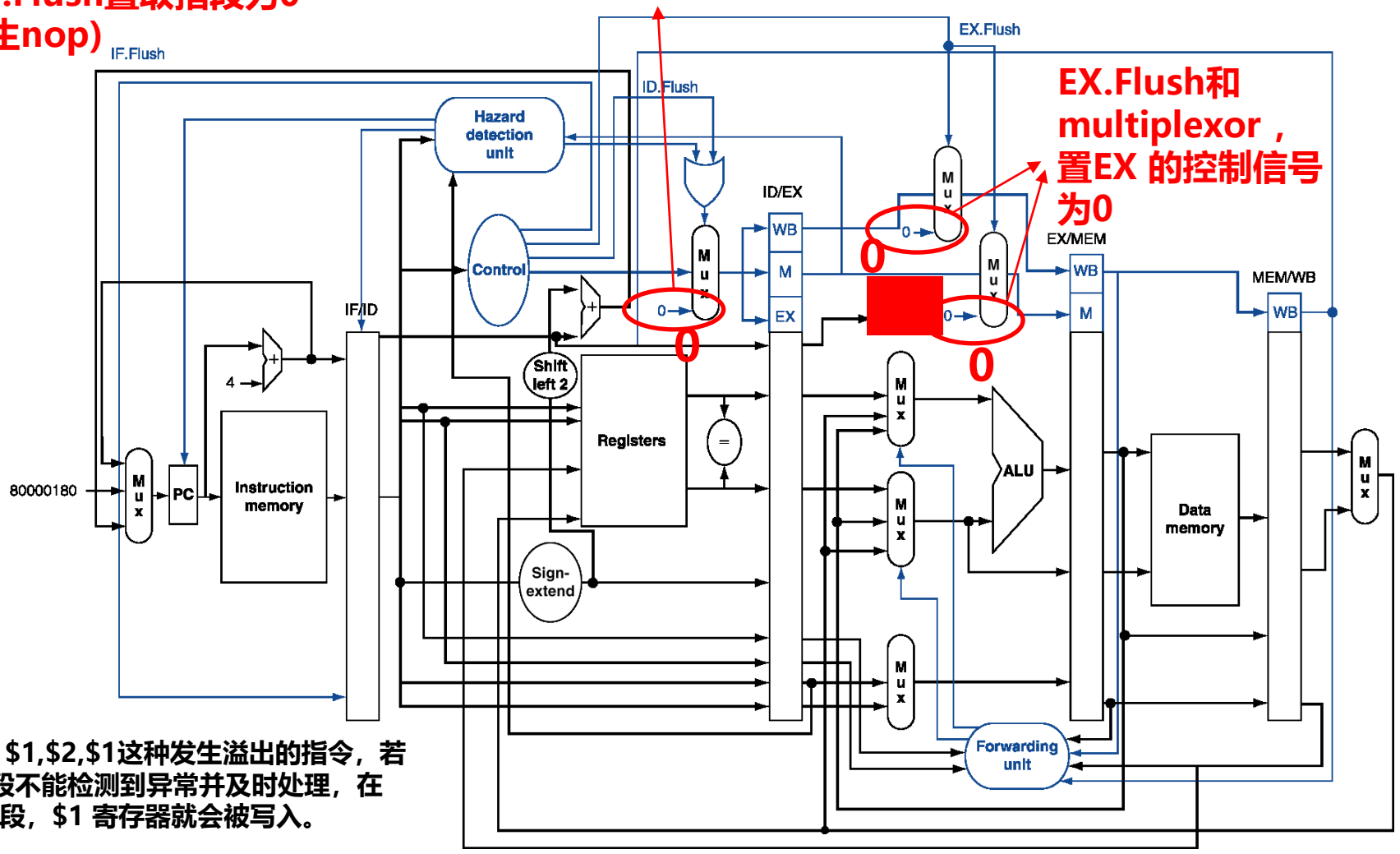
- **将处理异常的入口点写入PC(MIPS中处理溢出的地址为8000 0180)**

  (2位PCSource设置数据选择器，选择PC：PC+4、beq、Jump、8000 0180)

4.4 带有Flush和中断/异常处理的流水线处理器

IF.Flush置取指段为0
(产生nop)

ID.Flush 和冒险检测相或，置ID控制信号为0



EX.Flush和multiplexor，置EX 的控制信号为0

像add $1,$2,$1这种发生溢出的指令，若在EX 段不能检测到异常并及时处理，在后续的段，$1 寄存器就会被写入。

4.4 带有Flush和中断/异常处理的流水线处理器

对于下列程序，假设4C处的add出现溢出

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

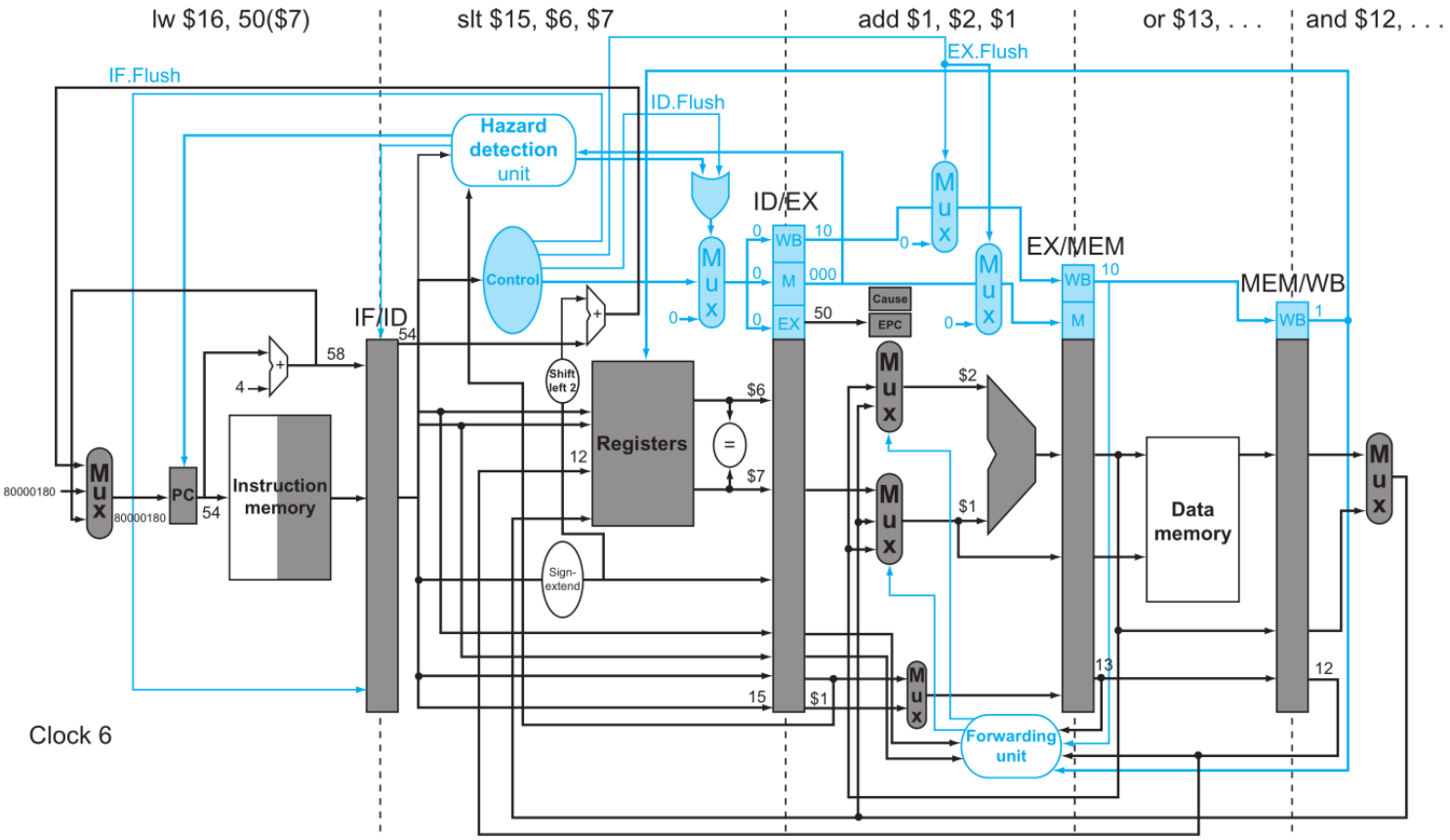
继而唤醒异常处理子程序：

```
80000180     sw    $26, 1000($0)
80000184     sw    $27, 1004($0)
...
```

流水线会发生什么?

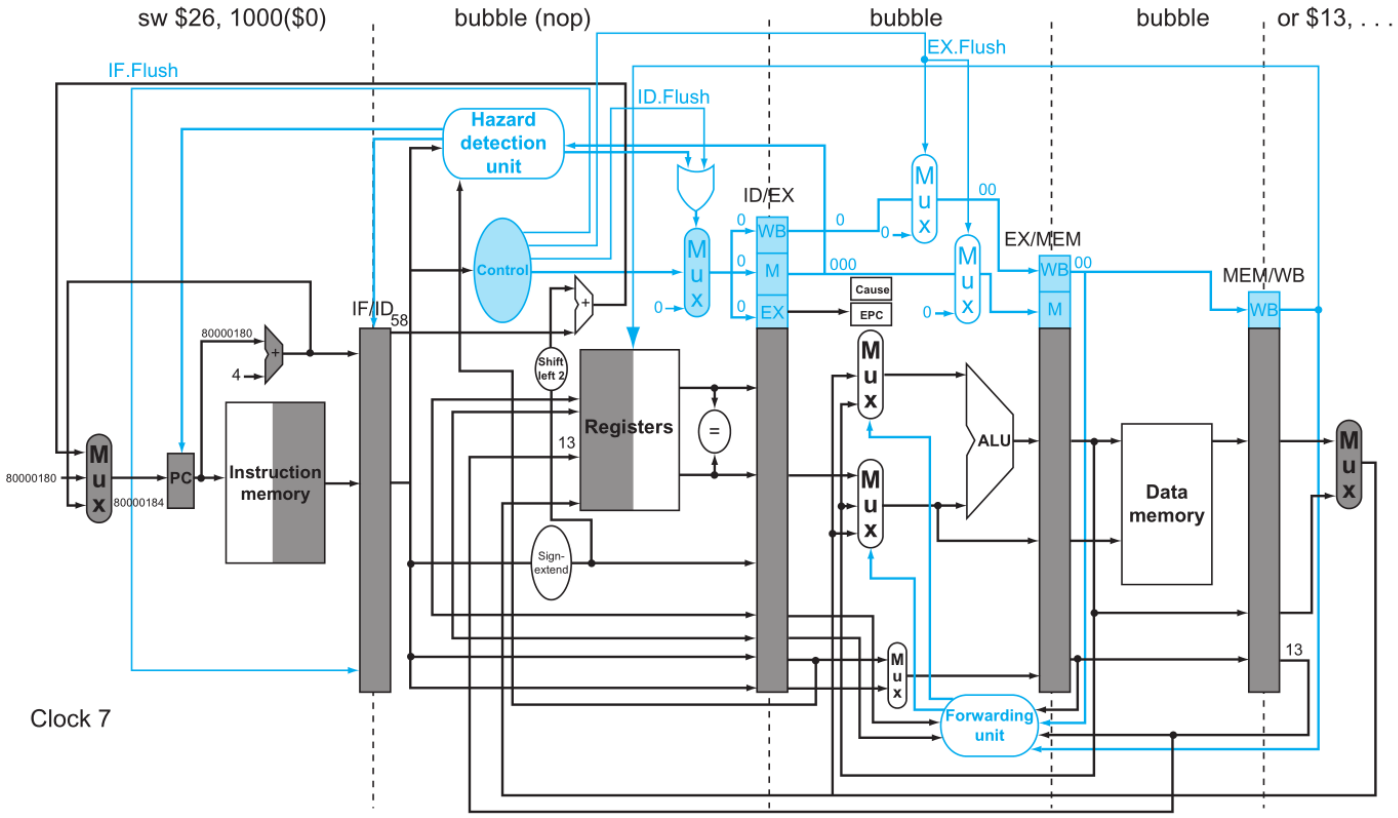
4.4 带有Flush和中断/异常处理的流水线处理器

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



The overflow is detected during the EX stage of clock 6, saving the address following the add in the EPC register (4C + 4 = 50 hex). Overflow causes all the Flush signals to be set near the end of this clock cycle, deasserting control values (setting them to 0) for the add.

4.4 带有Flush和中断/异常处理的流水线处理器



Clock cycle 7 shows the instructions converted to bubbles in the pipeline plus the fetching of the first instruction of the exception routine—sw $25,1000($0)—from instruction location 80000180_{hex} . Note that the and and or instructions, which are prior to the add, still complete. Although not shown, the ALU overflow signal is an input to the control unit.

4.4 中断/异常的优先级

问题：每个时钟周期会处理五条指令，如何处理同一周期内的多个异常？

->区分优先级进行处理。

MIPS 处理的策略是最早发生异常(中断)的指令优先处理

Cause register用一个”pending exception field”记录所有可能的异常信息，从而允许硬件在处理完先报的异常后，继续处理后续异常

- EPC捕获异常指令的地址，Cause register记录一个周期内所有可能的异常，继而由软件完成异常和指令的匹配，一个重要的线索在于：每个流水线级分别可以产生哪些异常？比如：“未定义的指令”会在ID阶段产生异常，而“系统调用”会在EX阶段。
- “I/O device request” 和“ 硬件出错”并不和某个指令相匹配，因此，处理这类异常比较灵活

4.4 操作系统与硬件的配合

硬件：

- 停止指令的执行
- 让所有在其前的指令执行完毕，Flush掉后面的指令
- 设置原因寄存器(Cause register)
- 将产生异常的指令位置保存于EPC
- 跳转到预先安排的异常处理程序的地址

操作系统：

- 查看异常原因，作适当调整：
  - 对”未定义的指令”、”硬件出错”或“算术溢出”异常：停止程序或进程，返回错误原因
  - 对“I/O device request ”或“系统调用”异常：保存程序的状态，执行期望的任务，以便在之后恢复程序的执行