

# 异常与中断

异常并不意外！——llxx😊

llxx@ustc.edu.cn

# 程序如何执行的？用到哪些设备？

```
/*---sum.c---*/
```

```
int sum(int a[ ], unsigned len)
```

```
{  
    int i, sum = 0;  
    for (i = 0; i <= len-1; i++)  
        sum += a[i];  
    return sum;  
}
```

```
/*---main.c---*/
```

```
int main()  
{  
    int a[1]={100};  
    int s;  
    s=sum(a,0);  
    printf("%d",s);  
}
```

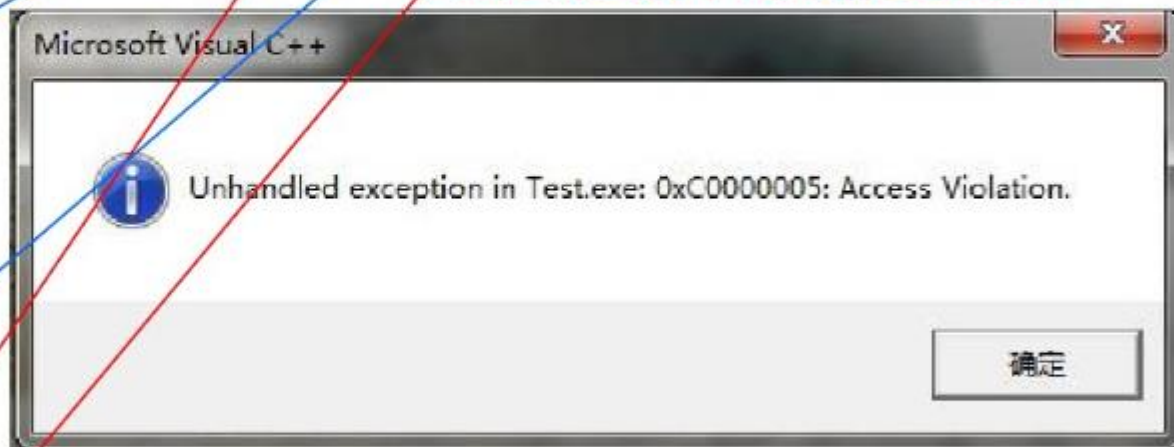
数据的表示

数据的运算

各类语句的转换与表示(指令)

各类复杂数据类型的转换表示

过程（函数）调用的转换表示



链接（linker）和加载

程序执行（存储器访问）

异常和中断处理

输入输出(I/O)

- C语言计算机？

IF

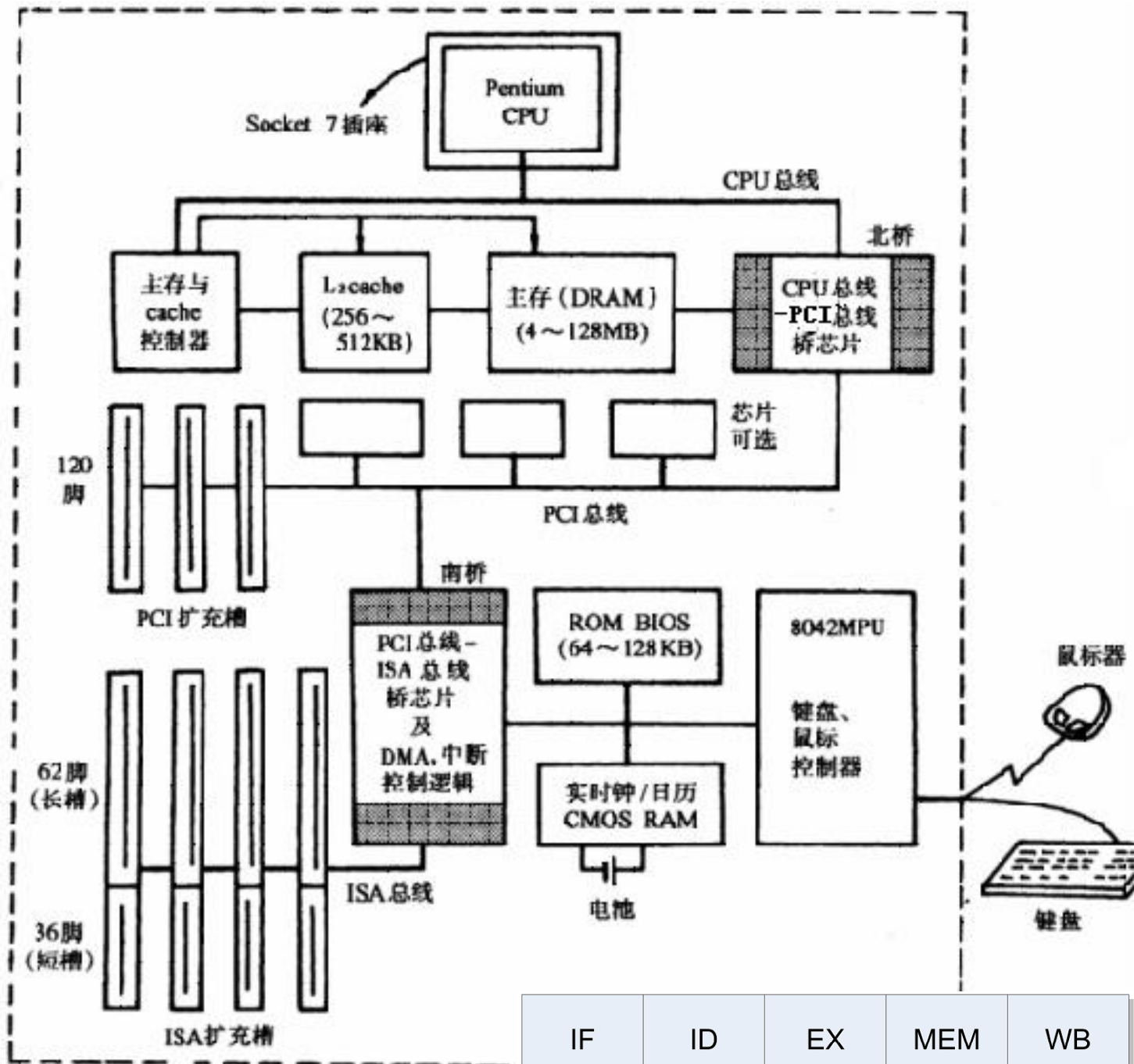
ID

EX

MEM

WB

# 输入输出

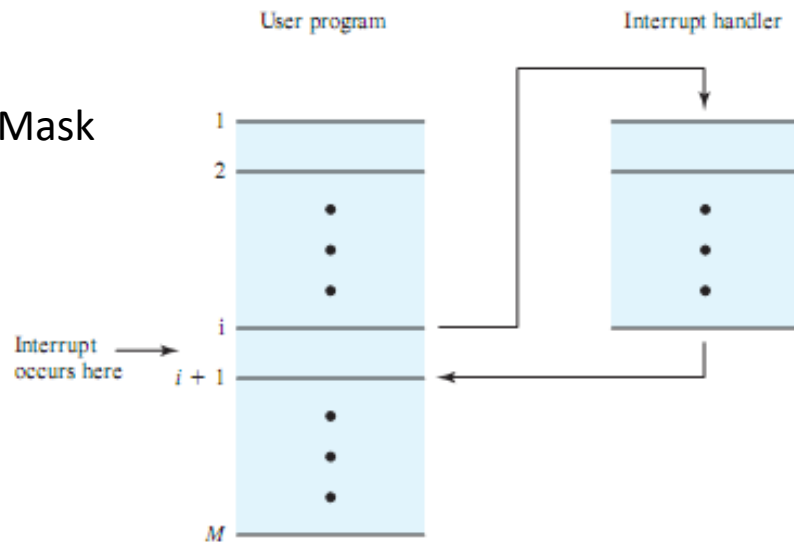
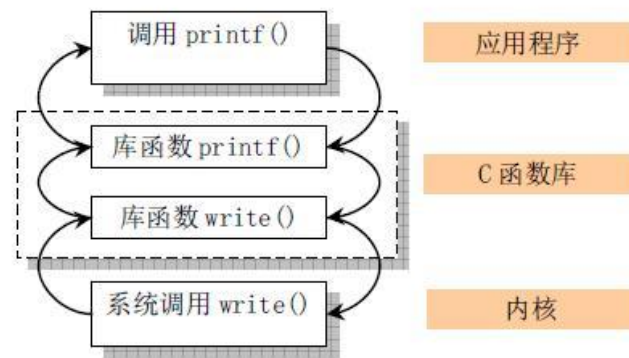


唐图10.19

|    |    |    |     |    |
|----|----|----|-----|----|
| IF | ID | EX | MEM | WB |
|----|----|----|-----|----|

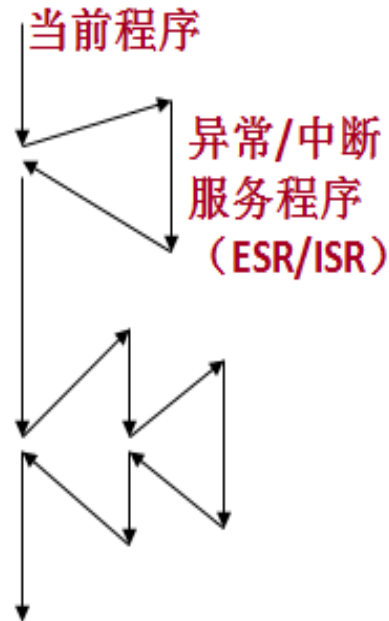
# 内容提要

- 异常与中断的基本概念，4.9
  - 功能：处理意外事件，I/O
    - 指令异常，系统服务（syscall）
    - 外部中断
  - 时序：顺序语义
    - 指令周期：同步，异步（中断周期）
  - 属性：5种
- 异常与中断响应的基本过程
  - 机构
    - 向量式/非向量式，控制栈
    - 异常：EPC, Cause, Vector
    - 中断：EPC, Cause, Vector, Enable, Mask
  - OS: ESR/ISR
  - 约定：软硬件接口
- 异常与中断响应过程控制
  - 多周期异常处理：中断周期
  - 流水线异常处理
    - 精确，非精确



# Terms: exceptions、interrupts、traps

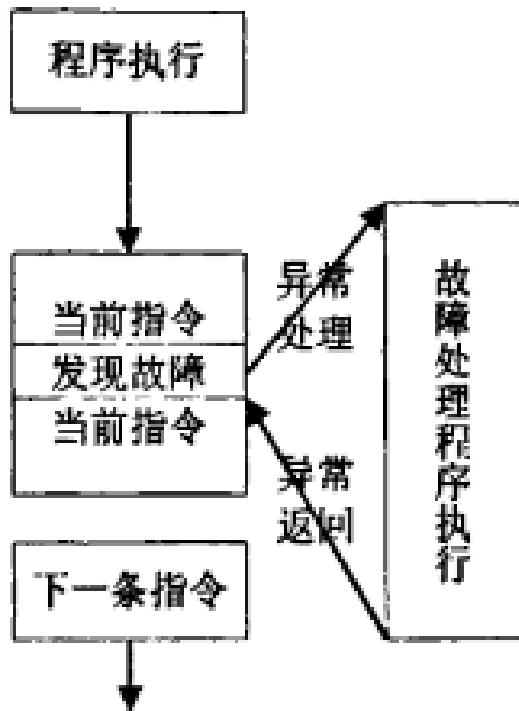
- 改变正常指令执行流（Transfer of Control）的**意外**事件。
  - 暂停当前程序的执行，转而执行其他程序；
  - 其他程序执行完成后，**恢复**被中断程序的执行。
  - 与**过程调用**的区别：发生场景（程序内/外），系统状态切换
- RISC: MIPS/ARM/RV统称exceptions，含
  - 内部事件
    - 指令异常：非法指令、溢出、访存错
    - 系统调用（syscall）：软中断（陷阱）、断点break
  - 外部事件：中断Interrupts
    - I/O中断（可屏蔽），设备故障（不可屏蔽）



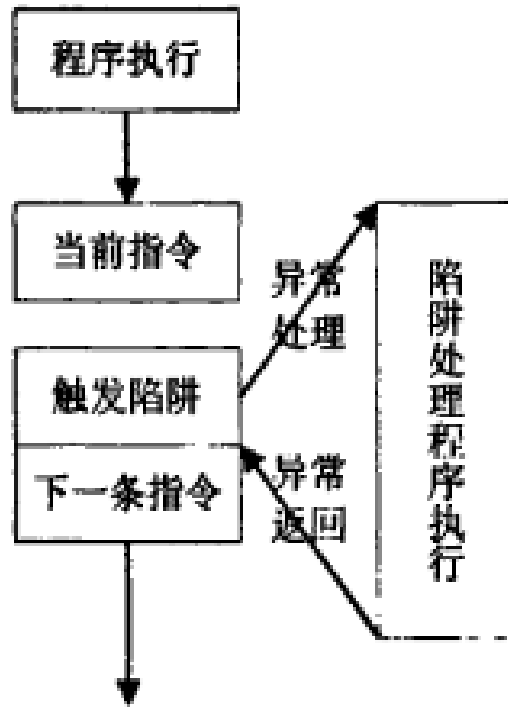
| Type of event                                 | From where? | RISC-V terminology |
|---|-------------|--------------------|
| System reset                                  | External    | Exception          |
| I/O device request                            | External    | Interrupt          |
| Invoke the operating system from user program | Internal    | Exception          |
| Using an undefined instruction                | Internal    | Exception          |
| Hardware malfunctions                         | Either      | Either             |

# 指令异常响应行为：同步

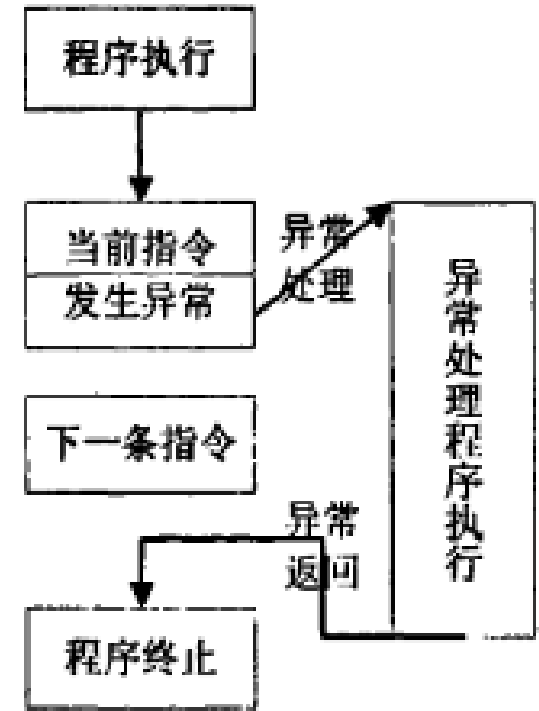
- 可恢复异常：当前指令故障，OS处理后返回当前指令，如图“故障”
- 不可恢复异常：当前异常无法修复，交用户处理，如图“异常中止”
- 陷阱：由陷阱指令（软中断）触发调用系统服务，返回下一条指令



故障处理

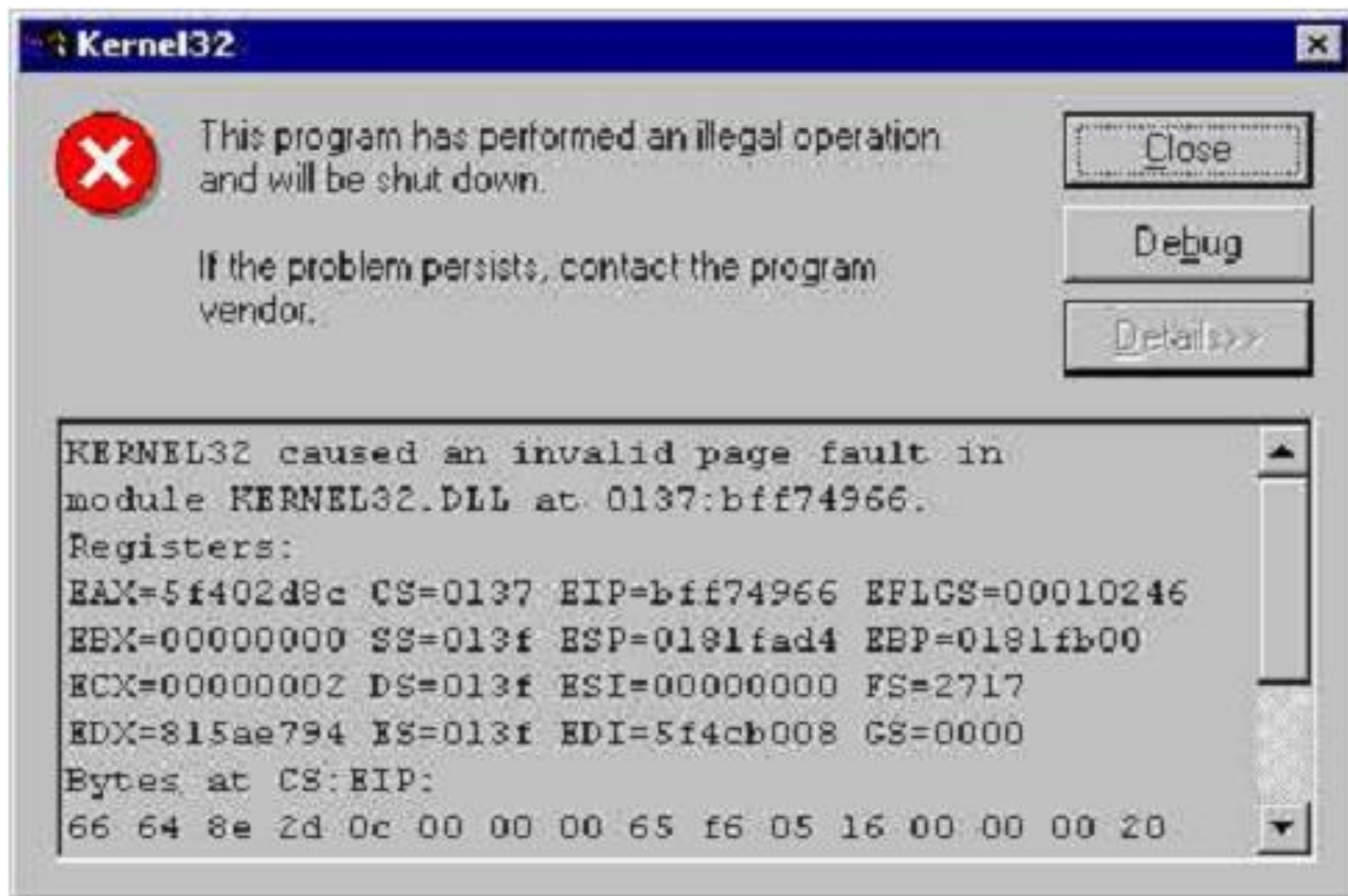


陷阱处理

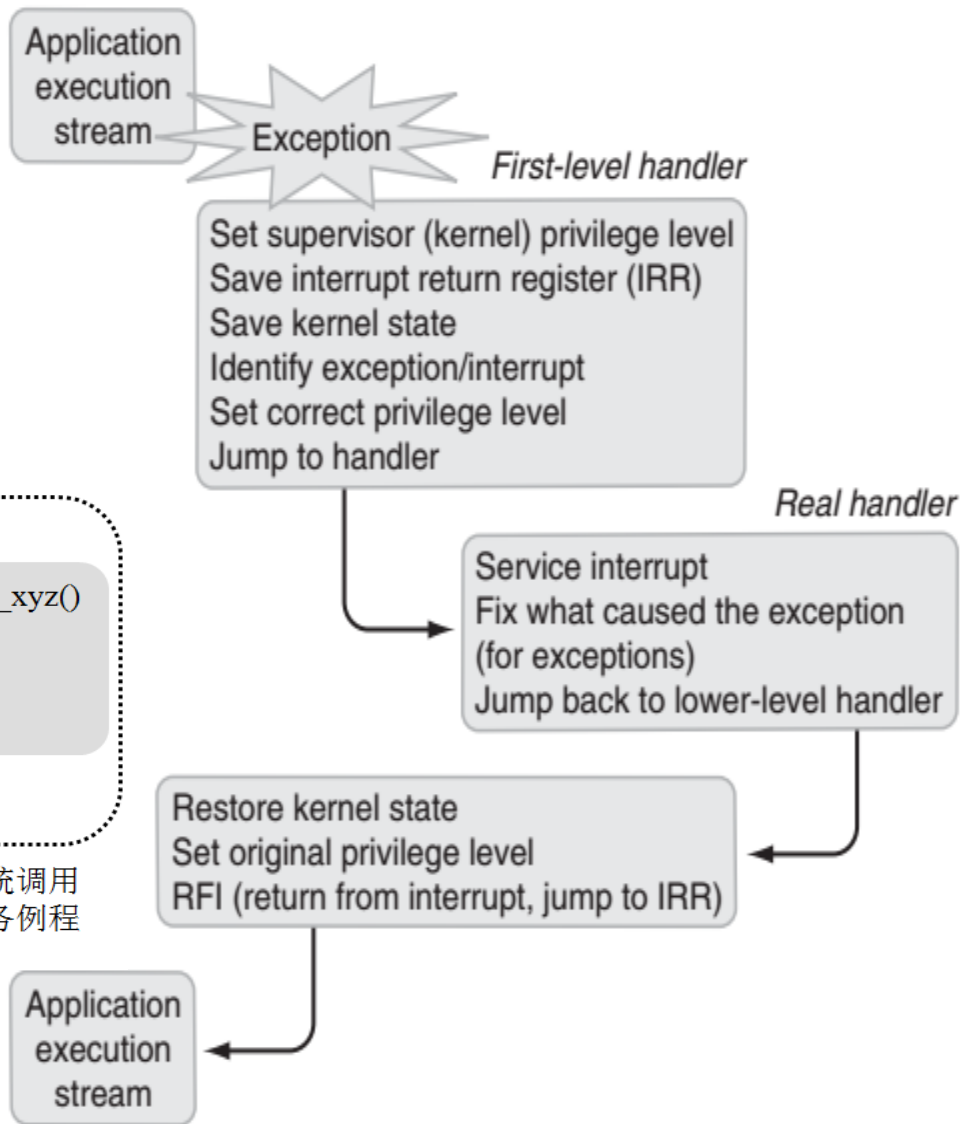
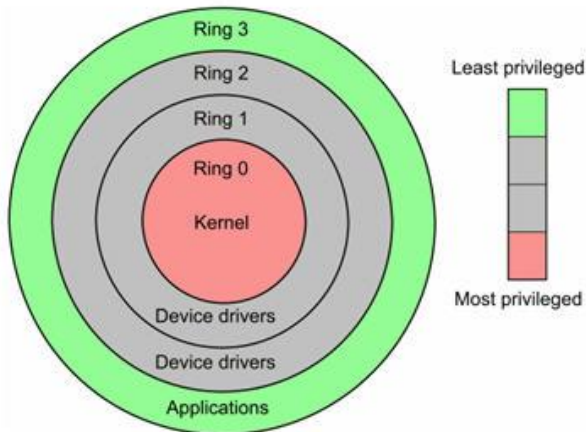


异常中止

# 不可恢复异常：异常终止

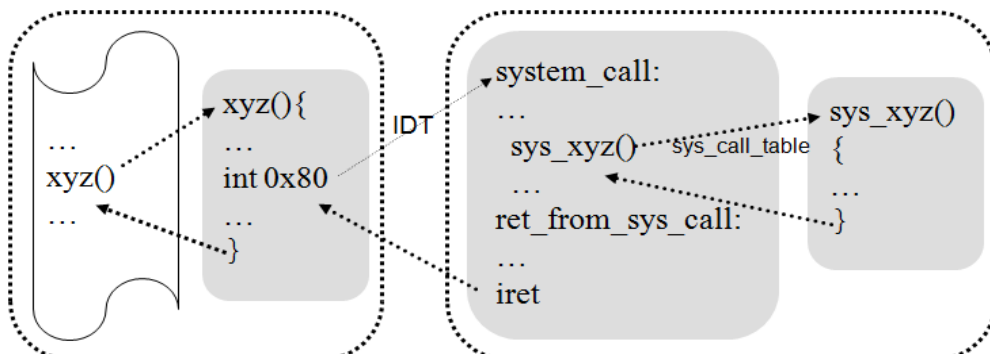


# 系统调用：应用程序调用OS服务



用户态

内核态



在应用程序调用中的系统调用

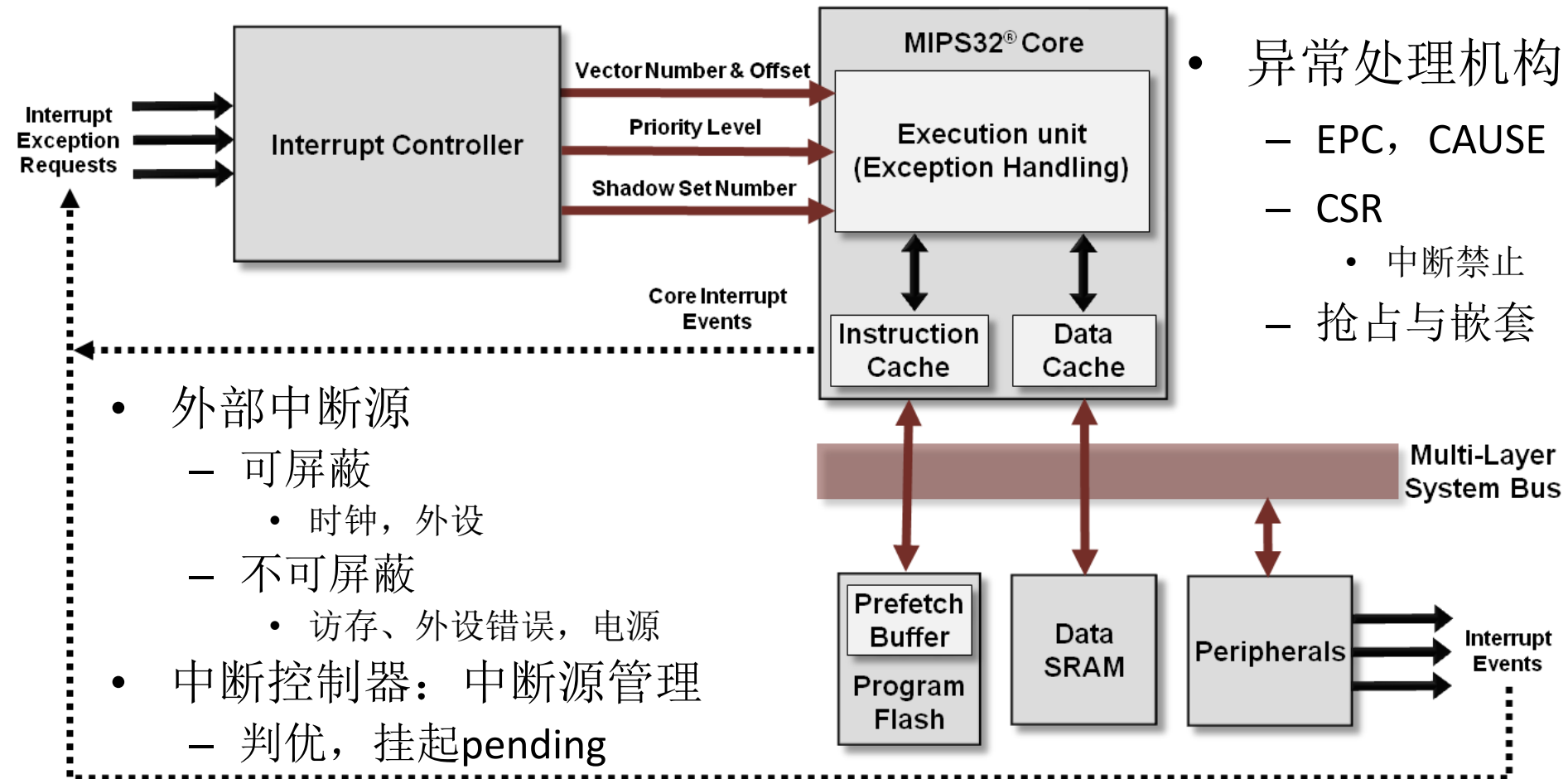
在libc标准库中的封装例程

系统调用处理程序

系统调用服务例程



# 外部中断响应行为：异步

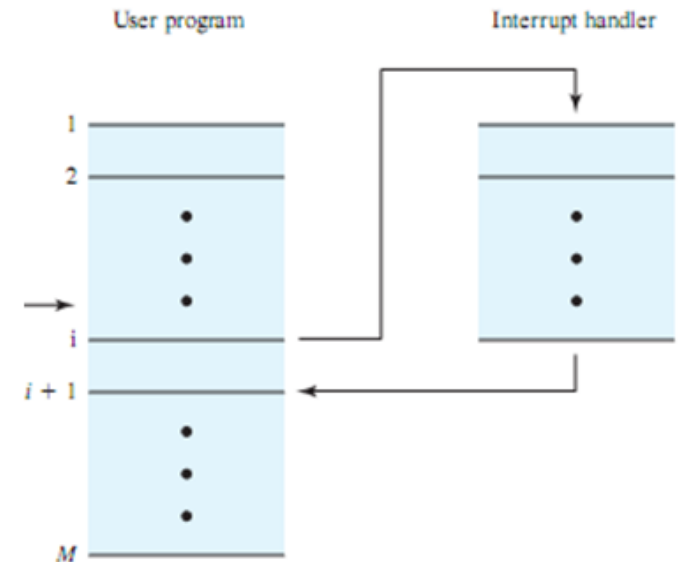
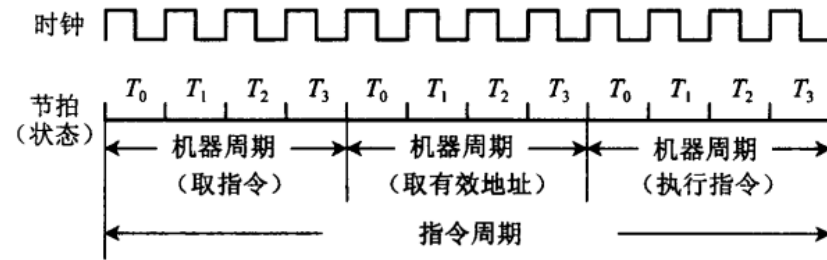
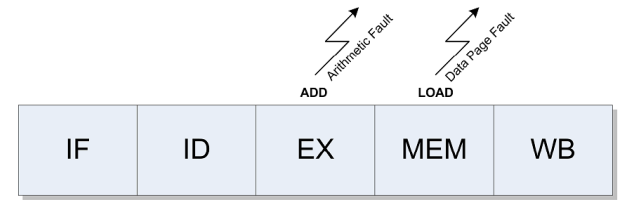


# RISC异常/中断的属性：5种

| Exception type                                  | Synchronous vs. asynchronous | User request vs. coerced | User maskable vs. nonmaskable | Within vs. between instructions | Resume vs. terminate |
|---|------------------------------|--------------------------|-------------------------------|---------------------------------|----------------------|
| I/O device request                              | Asynchronous                 | Coerced                  | Nonmaskable                   | Between                         | Resume               |
| Invoke operating system                         | Synchronous                  | User request             | Nonmaskable                   | Between                         | Resume               |
| Tracing instruction execution                   | Synchronous                  | User request             | User maskable                 | Between                         | Resume               |
| Breakpoint                                      | Synchronous                  | User request             | User maskable                 | Between                         | Resume               |
| Integer arithmetic overflow                     | Synchronous                  | Coerced                  | User maskable                 | Within                          | Resume               |
| Floating-point arithmetic overflow or underflow | Synchronous                  | Coerced                  | User maskable                 | Within                          | Resume               |
| Page fault                                      | Synchronous                  | Coerced                  | Nonmaskable                   | Within                          | Resume               |
| Misaligned memory accesses                      | Synchronous                  | Coerced                  | User maskable                 | Within                          | Resume               |
| Memory protection violations                    | Synchronous                  | Coerced                  | Nonmaskable                   | Within                          | Resume               |
| Using undefined instructions                    | Synchronous                  | Coerced                  | Nonmaskable                   | Within                          | Terminate            |
| Hardware malfunctions                           | Asynchronous                 | Coerced                  | Nonmaskable                   | Within                          | Terminate            |
| Power failure                                   | Asynchronous                 | Coerced                  | Nonmaskable                   | Within                          | Terminate            |

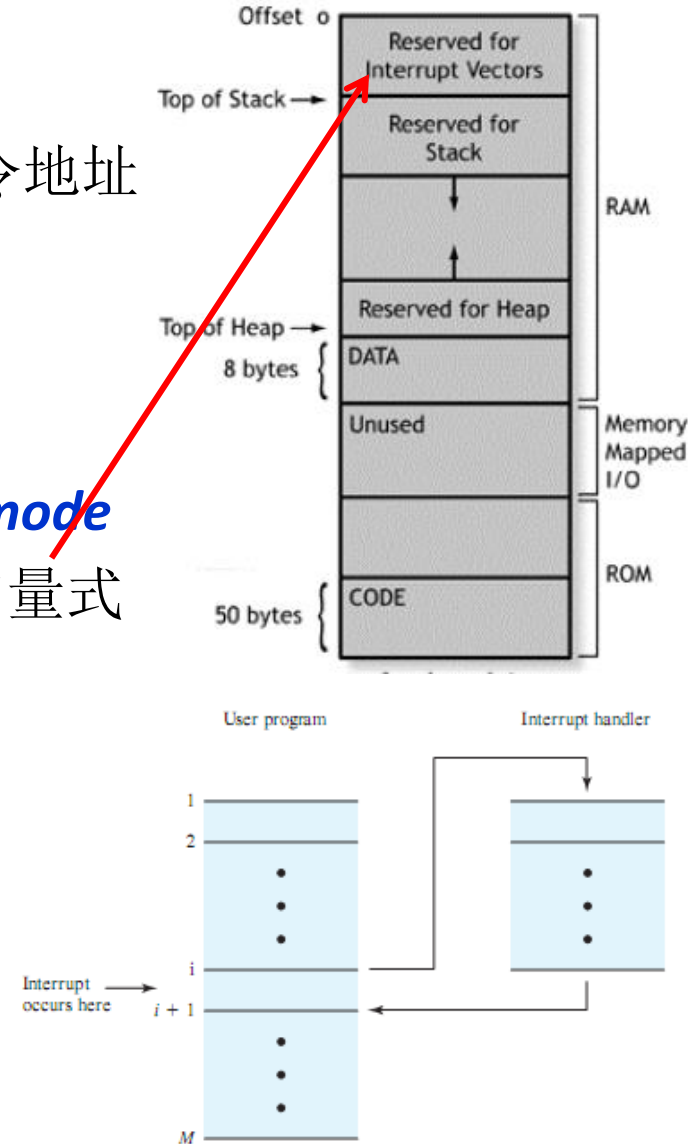
# 异常、中断的**发生时刻**与**响应时刻**

- 异常：立即响应（同步）
  - 指令异常：终止执行，或重启当前指令
  - 系统调用：**返回**下一条指令
- 中断：指令周期结束才响应
  - **返回**下一条指令，停止
- 精确断点： p228
  - 之前的指令已执行完成
    - 已经提交其**体系结构可见状态**
  - 之后的指令还没有发射
    - 没有改变任何机器状态
- 程序语义：顺序，按序，精确



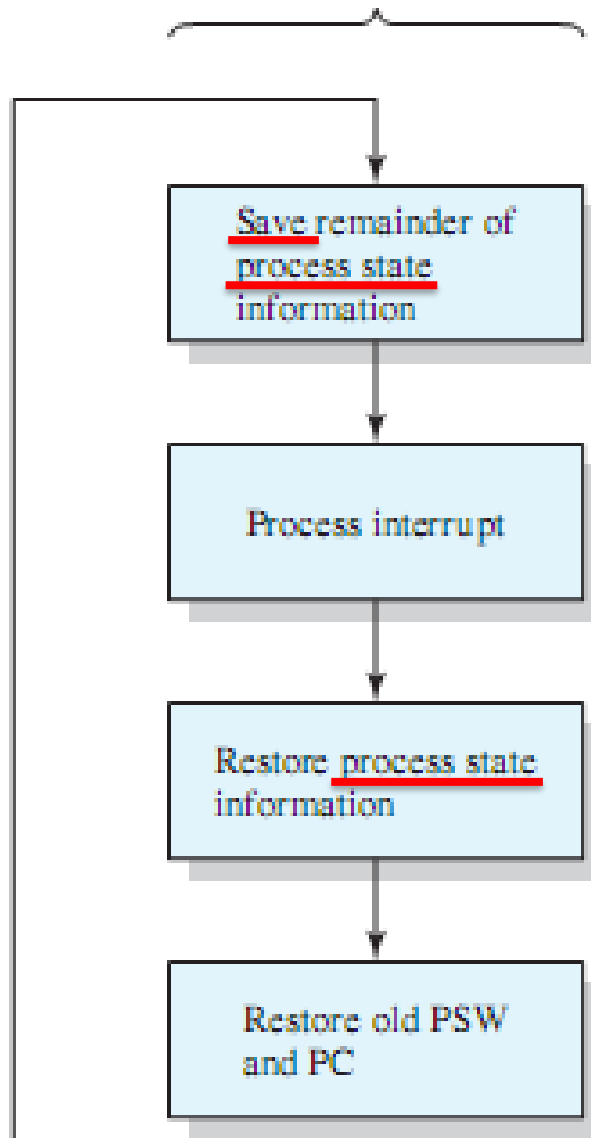
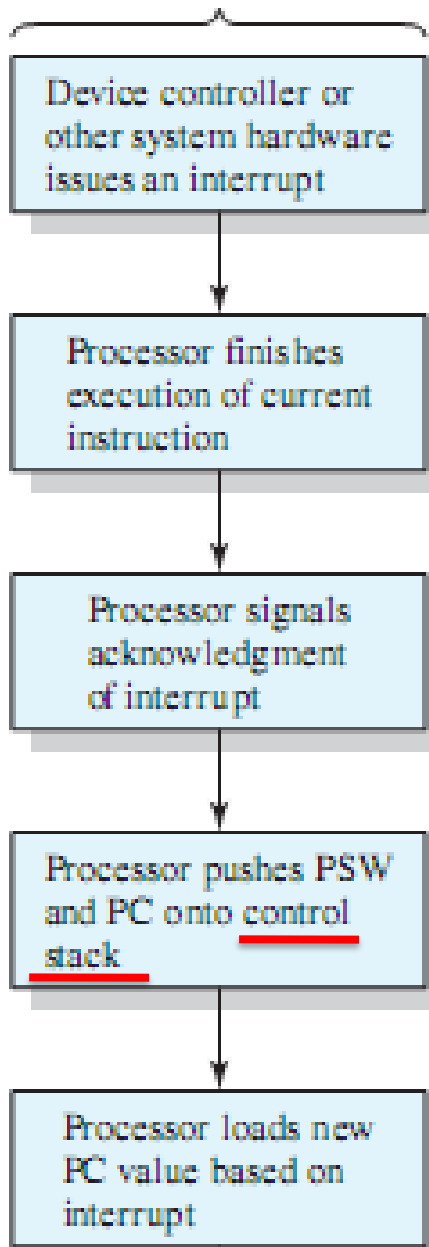
# The Basics of Exception Handling

- 软硬件接口约定 (p228)
  - EPC: 断点=异常指令地址/下一条指令地址
  - Cause: 检测并记录异常原因
  - 关中断, 清当前中断
  - 保存PSW/CSR
  - 建立处理环境: *user mode => kernel mode*
  - Vector: 转ESR/ISR入口, 非向量式/向量式
    1. 保存现场: RF
    2. 开中断
    3. 进行异常服务
    4. 关中断
    5. 恢复现场: RF
    6. 中断返回: iret/eret
      - 开中断, 恢复用户模式, 恢复PSW

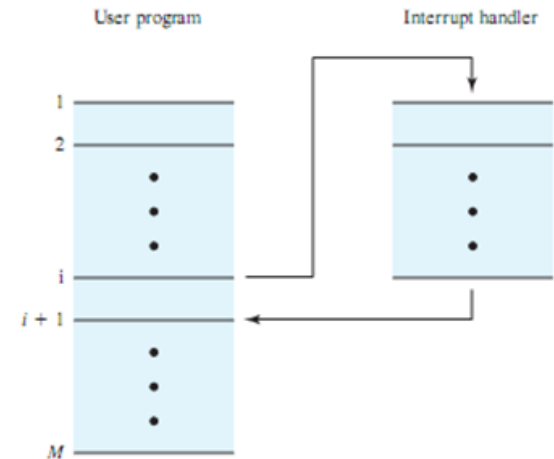


## Hardware

## Software

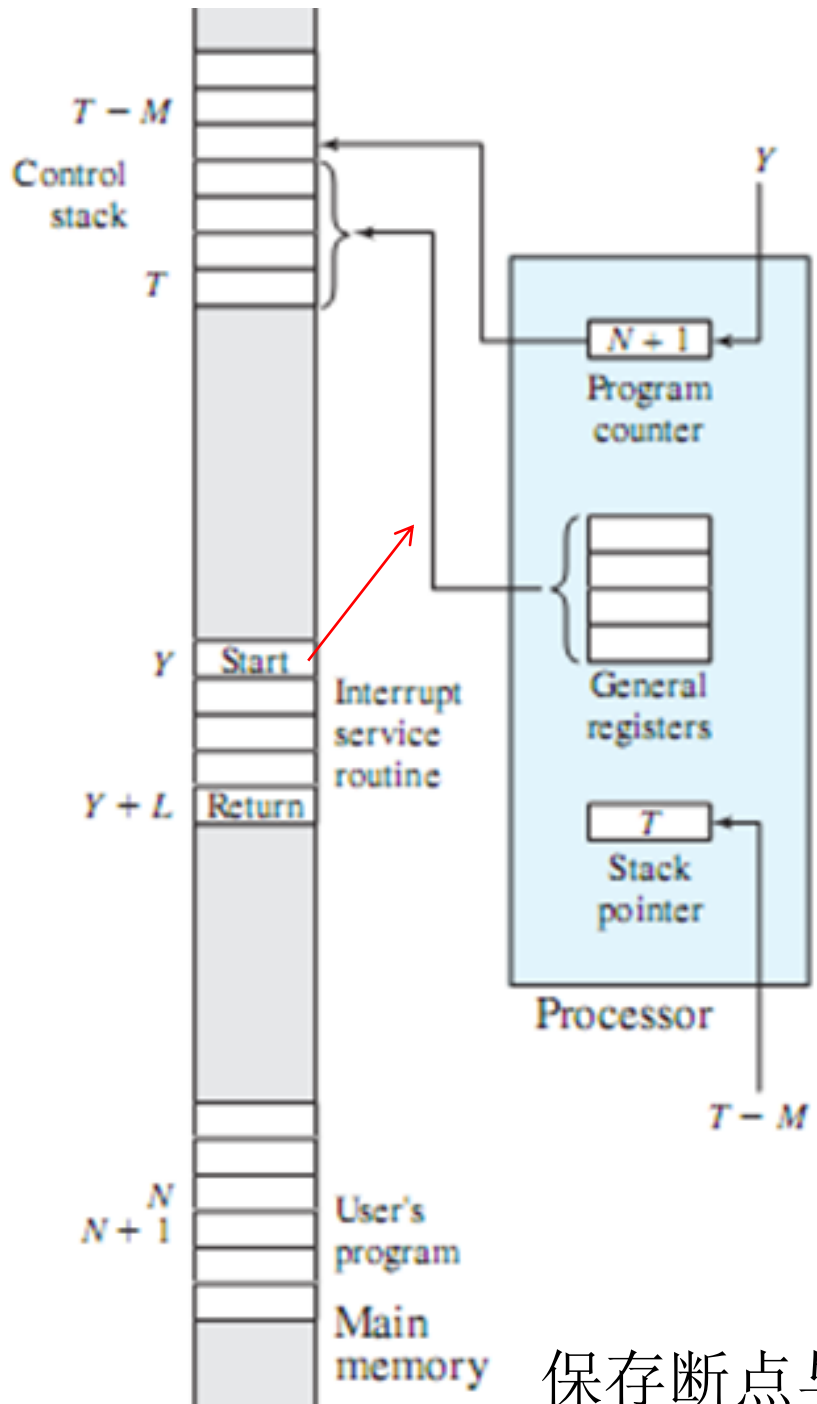


# Simple Excp/Intr Processing 软硬件接口约定

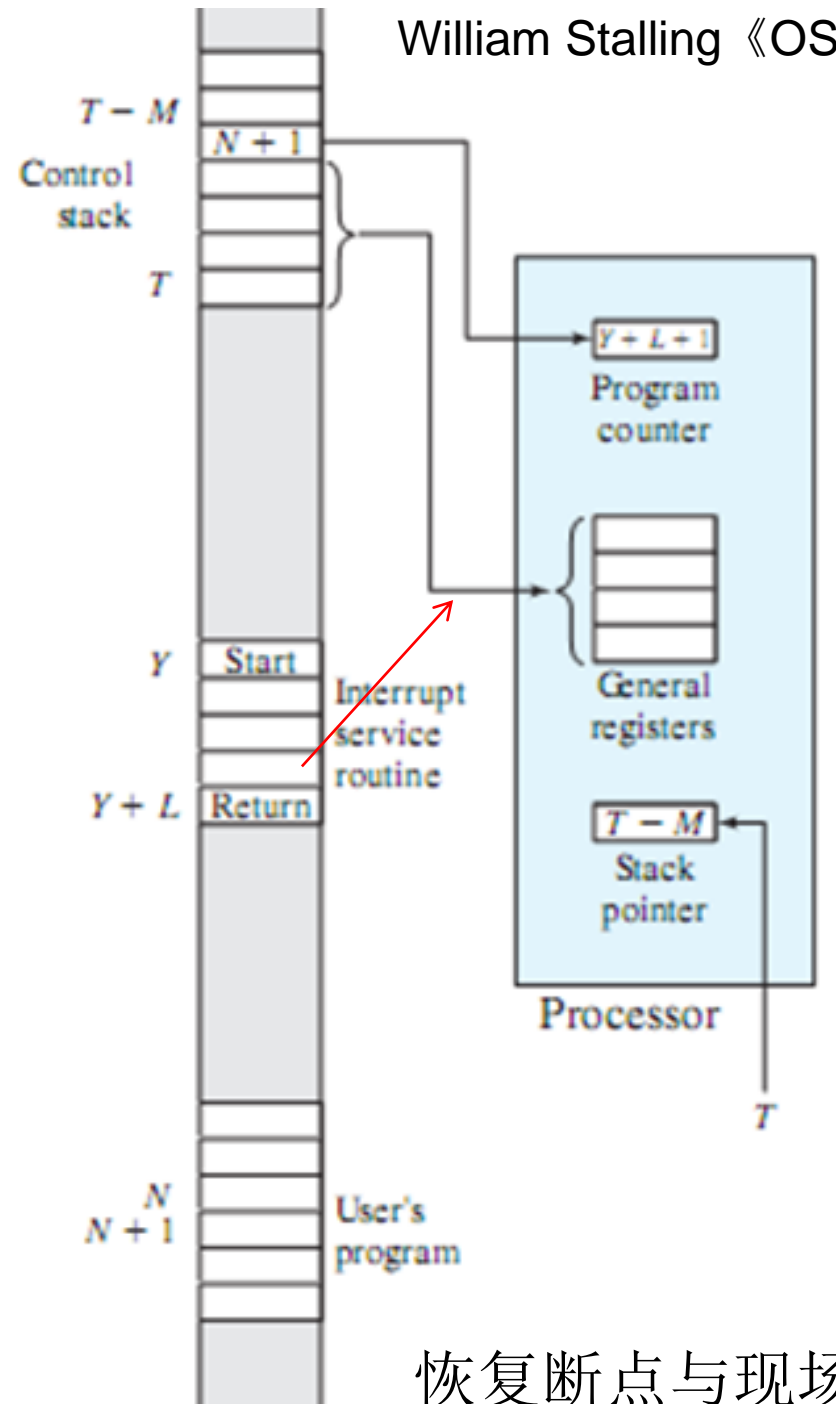


RV\$4.9.1 ISR入口地址

| Exception type                      | Exception vector address to be added to a Vector Table Base Register |
|-------------------------------------|--|
| Undefined instruction               | 00 0100 0000 <sub>two</sub>  |
| System Error (hardware malfunction) | 01 1000 0000 <sub>two</sub>  |



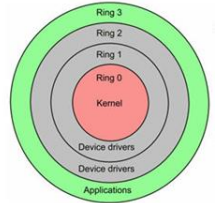
保存断点与现场



恢复断点与现场

# RV Control and Status Register

- RV工作模式: user mode, supervisor mode, machine mode
- CSR: 4K独立的MMIO地址空间
  - 不同工作模式各有一套CSR
    - 公共: Status, EPC, Cause, *Scratch* (用于模式切换) . . .
  - 读写模式: *read-modify-write cycle*

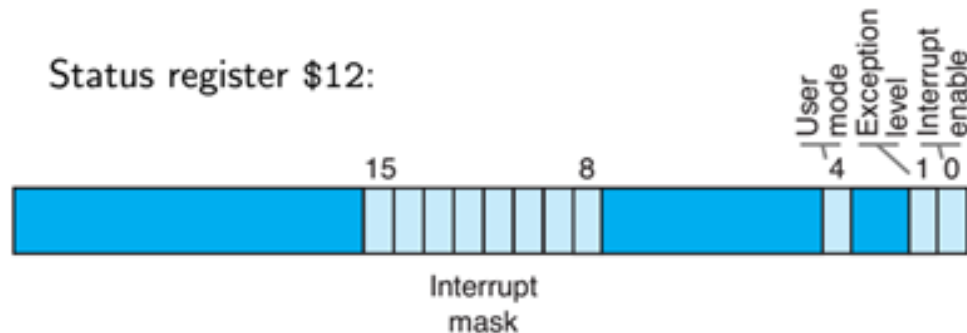


| Number             | Privilege | Name     | Description                              |
|--------------------|-----------|----------|--|
| User Trap Setup    |           |          |  |
| 0x000              | URW       | ustatus  | User status register.                    |
| 0x004              | URW       | uie      | User interrupt- <u>enable</u> register.  |
| 0x005              | URW       | utvec    | User trap handler base address.          |
| User Trap Handling |           |          |  |
| 0x040              | URW       | uscratch | Scratch register for user trap handlers. |
| 0x041              | URW       | uepc     | User exception program counter.          |
| 0x042              | URW       | ucause   | User trap cause.                         |
| 0x043              | URW       | ubadaddr | User bad address.                        |
| 0x044              | URW       | uip      | User interrupt <u>pending</u> .          |

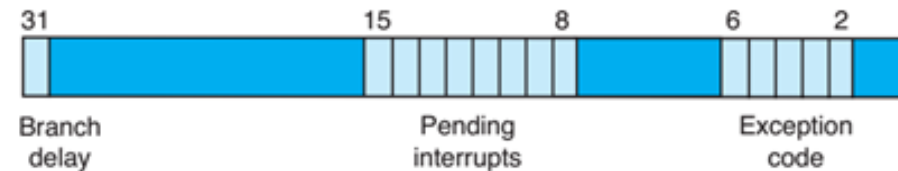
User mode CSR

# Important Exceptions registers

Status register \$12:



Cause register \$13:



| Number | Name | Cause of exception                                  |
|--------|------|---|
| 0      | Int  | interrupt (hardware)                                |
| 4      | AdEL | address error exception (load or instruction fetch) |
| 5      | AdES | address error exception (store)                     |
| 6      | IBE  | bus error on instruction fetch                      |
| 7      | DBE  | bus error on data load or store                     |
| 8      | Sys  | syscall exception                                   |
| 9      | Bp   | breakpoint exception                                |
| 10     | RI   | reserved instruction exception                      |
| 11     | CpU  | coprocessor unimplemented                           |
| 12     | Ov   | arithmetic overflow exception                       |
| 13     | Tr   | trap  |
| 15     | FPE  | floating point                                      |



# 访问CSR, §5.14

- CSR swap instructions
  - 6条: I-type, 12位=4K
  - 将CSR复制到通用Reg, 用寄存器操作数或Imm改写它们。

RV图5-47 (Fig 5-48)

| Type       | Mnemonic | Name                     |
|------------|----------|--------------------------|
| CSR Access | CSRRWI   | CSR Read/Write Immediate |
|            | CSRRSI   | CSR Read/Set Immediate   |
|            | CSRRCI   | CSR Read/Clear Immediate |
|            | CSRRW    | CSR Read/Write           |
|            | CSRRS    | CSR Read/Set             |
|            | CSRRC    | CSR Read/Clear           |

《The RISC-V Instruction Set Manual》

| 31          | 20 19 | 15 14     | 12 11  | 7 6  | 0      |
|-------------|-------|-----------|--------|------|--------|
| csr         |       | rs1       | funct3 | rd   | opcode |
| 12          |       | 5         | 3      | 5    | 7      |
| source/dest |       | source    | CSRRW  | dest | SYSTEM |
| source/dest |       | source    | CSRRS  | dest | SYSTEM |
| source/dest |       | source    | CSRRC  | dest | SYSTEM |
| source/dest |       | uimm[4:0] | CSRRWI | dest | SYSTEM |
| source/dest |       | uimm[4:0] | CSRRSI | dest | SYSTEM |
| source/dest |       | uimm[4:0] | CSRRCI | dest | SYSTEM |

# ISR编程：RV Arch'ed Regs

| Name    | Register number | Usage                          | Preserved on call? |
|---------|-----------------|--------------------------------|--------------------|
| x0      | 0               | The constant value 0           | n.a.               |
| x1 (ra) | 1               | Return address (link register) | yes                |
| x2 (sp) | 2               | Stack pointer                  | yes                |
| x3 (gp) | 3               | Global pointer                 | yes                |
| x4 (tp) | 4               | Thread pointer                 | yes                |
| x5-x7   | 5-7             | Temporaries                    | no                 |
| x8-x9   | 8-9             | Saved                          | yes                |
| x10-x17 | 10-17           | Arguments/results              | no                 |
| x18-x27 | 18-27           | Saved                          | yes                |
| x28-x31 | 28-31           | Temporaries                    | no                 |

RV 图2-14

过程调用时可用a0~a7, s0~s11, t0~t6

ABI calling convention: ——减少代码量

- **a/t**-registers are caller-saved
- **s**-regs are callee-saved and **preserve** their contents across function calls

| Register | ABI Name | Description                       | Saver  |
|----------|----------|-----------------------------------|--------|
| x0       | zero     | Hard-wired zero                   | —      |
| x1       | ra       | Return address                    | Caller |
| x2       | sp       | Stack pointer                     | Callee |
| x3       | gp       | Global pointer                    | —      |
| x4       | tp       | Thread pointer                    | —      |
| x5       | t0       | Temporary/alternate link register | Caller |
| x6-7     | t1-2     | Temporaries                       | Caller |
| x8       | s0/fp    | Saved register/frame pointer      | Callee |
| x9       | s1       | Saved register                    | Callee |
| x10-11   | a0-1     | Function arguments/return values  | Caller |
| x12-17   | a2-7     | Function arguments                | Caller |
| x18-27   | s2-11    | Saved registers                   | Callee |
| x28-31   | t3-6     | Temporaries                       | Caller |

Table 25.1

The RISC-V Instruction Set Manual Volume I

# TimerISR

非向量式中断

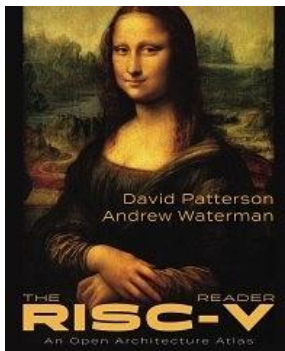
```
# save registers
csrrw a0, mscratch, a0 # save a0; set a0 = &temp storage
sw a1, 0(a0) # save a1
sw a2, 4(a0) # save a2
sw a3, 8(a0) # save a3
sw a4, 12(a0) # save a4

# decode interrupt cause
csrr a1, mcause # read exception cause
bgez a1, exception # branch if not an interrupt
andi a1, a1, 0x3f # isolate interrupt cause
li a2, 7 # a2 = timer interrupt cause
bne a1, a2, otherInt # branch if not a timer interrupt

# handle timer interrupt by incrementing time comparator
la a1, mtimecmp # a1 = &time comparator
lw a2, 0(a1) # load lower 32 bits of comparator
lw a3, 4(a1) # load upper 32 bits of comparator
addi a4, a2, 1000 # increment lower bits by 1000 cycles
sltu a2, a4, a2 # generate carry-out
add a3, a3, a2 # increment upper bits
sw a3, 4(a1) # store upper 32 bits
sw a4, 0(a1) # store lower 32 bits

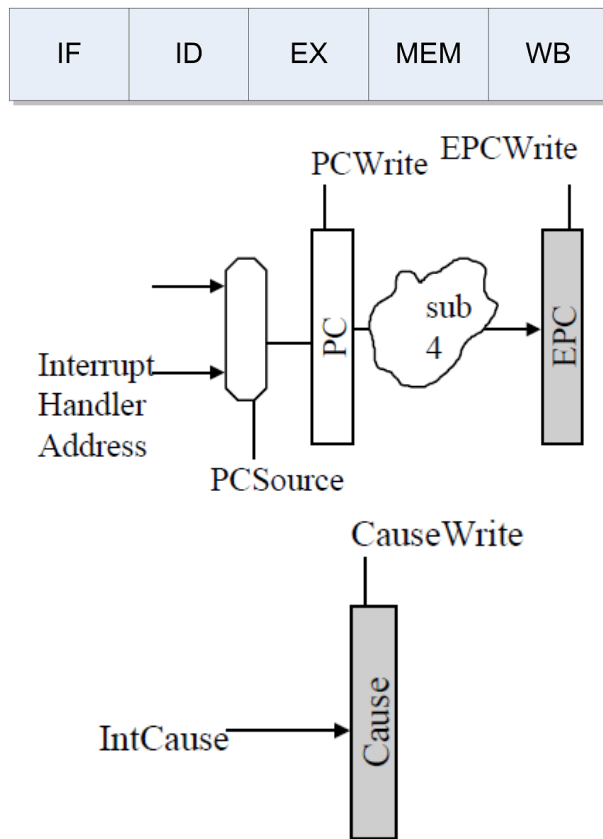
# restore registers and return
lw a4, 12(a0) # restore a4
lw a3, 4(a0) # restore a3
lw a2, 4(a0) # restore a2
lw a1, 0(a0) # restore a1
csrrw a0, mscratch, a0 # restore a0; mscratch = &temp storage
mret # return from handler
```

图10.6



# 指令异常实现：五步法

- 异常处理过程分解、定义微操作
  - 保存断点：PC-4 => EPC
  - 保存异常原因：Cause寄存器
    - 非法指令：在ID周期
    - 算术溢出：在EXE周期
    - 缺页
  - 转ESR：入口00 0010 0000，非向量式
  - ESR@OS
    - 根据cause分别处理
      - 非法指令：为用户程序提供某些服务
      - 溢出：报错
    - 返回：ERET (return-from-environment)
- 数据通路、控制信号、综合FSM
  - EPC, Cause, 转Vec
    - IntCause: 0-非法指令, 1-溢出
  - EPCWrite, CauseWrite, 分支

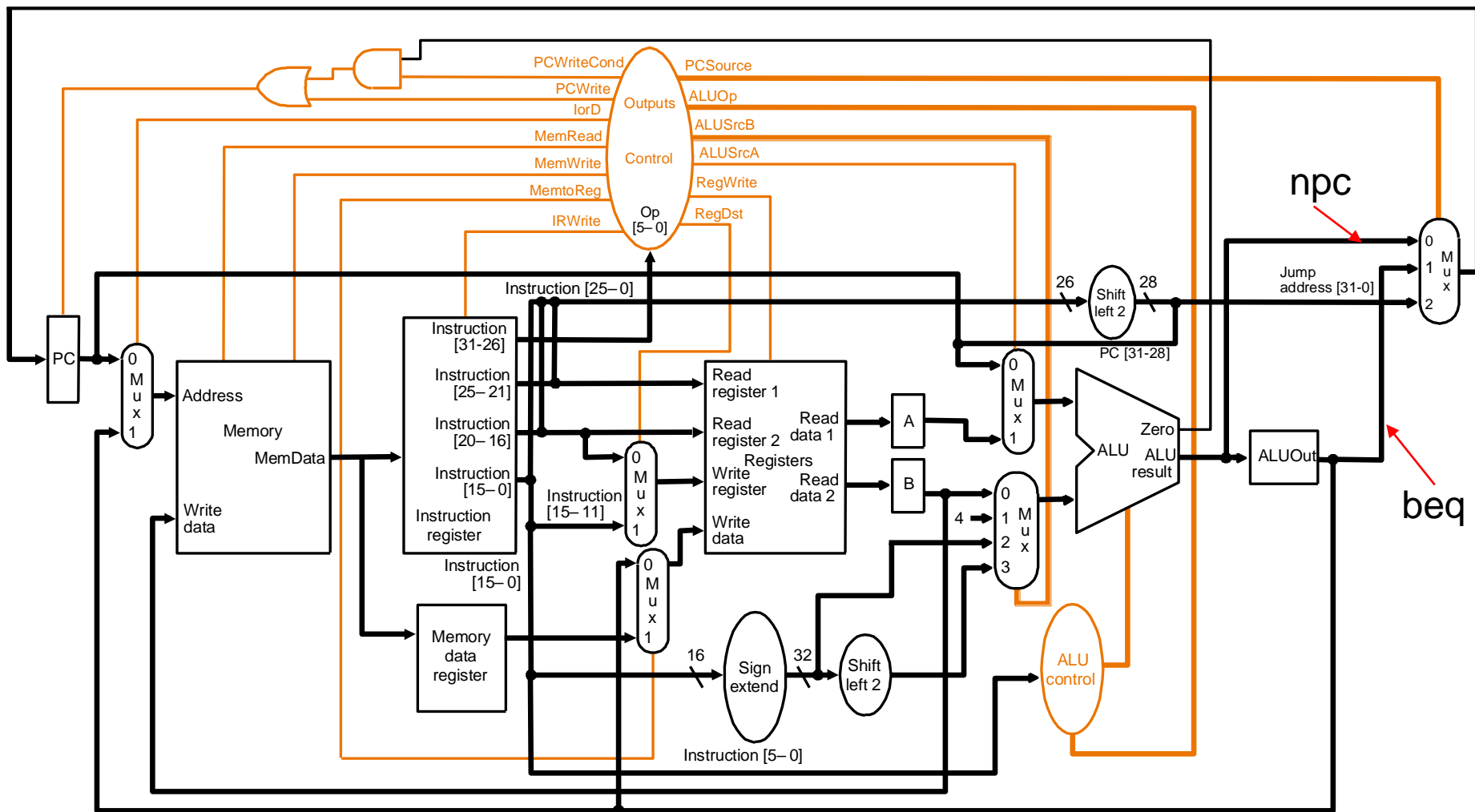


RV\$4.9.1 ESR/ISR入口地址

| Exception type                      | Exception vector address to be added to a Vector Table Base Register |
|-------------------------------------|--|
| Undefined instruction               | 00 0100 0000 <sub>two</sub>  |
| System Error (hardware malfunction) | 01 1000 0000 <sub>two</sub>  |

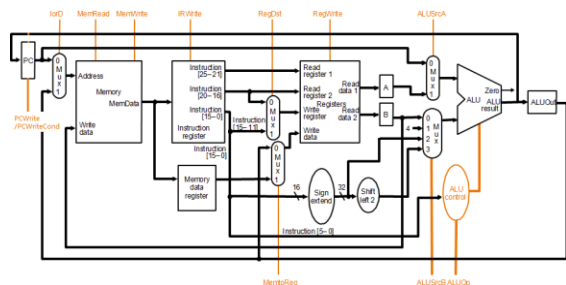
# 多周期总图

|    |    |    |     |    |
|----|----|----|-----|----|
| IF | ID | EX | MEM | WB |
|----|----|----|-----|----|

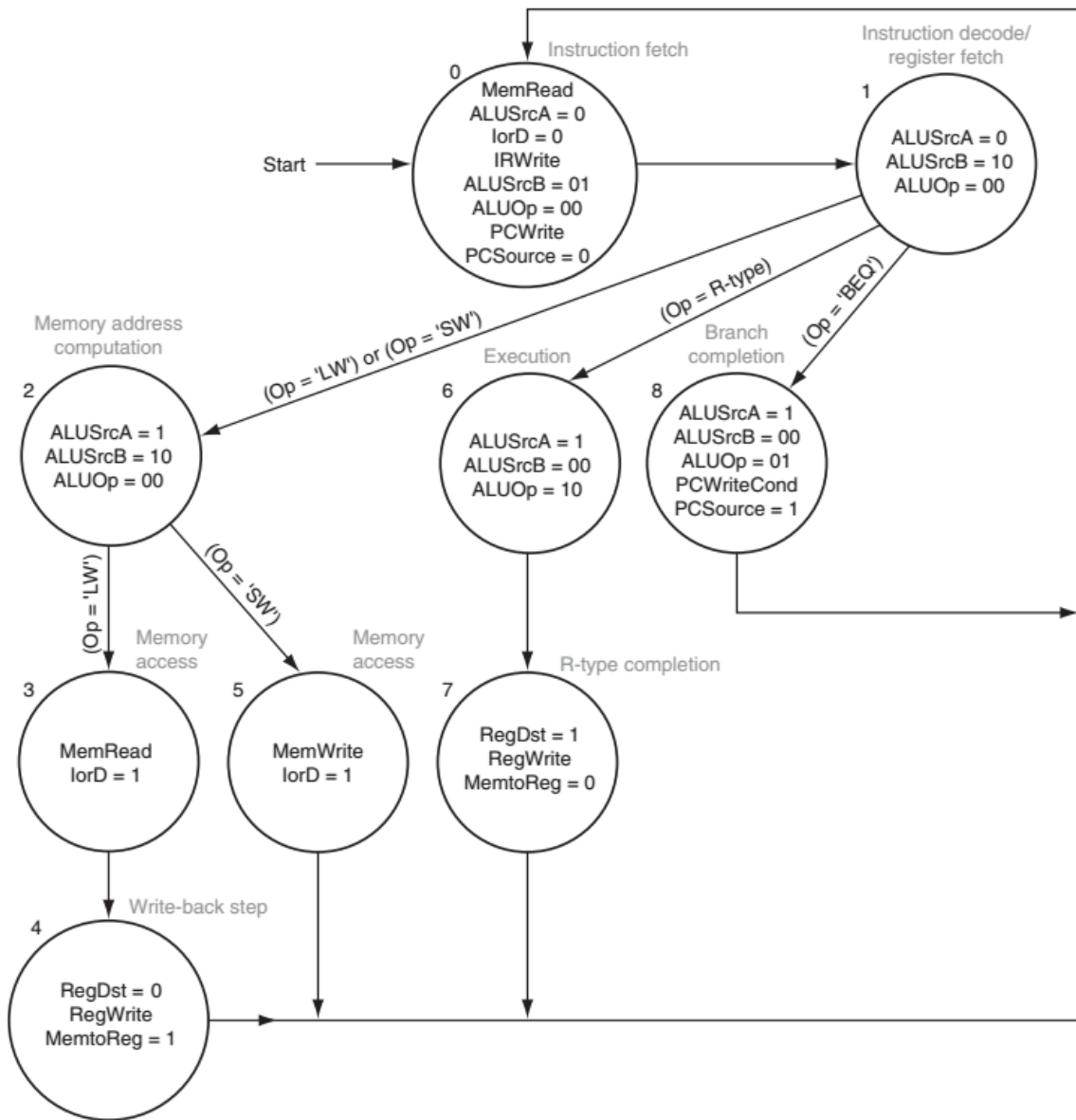
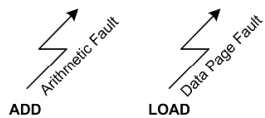


# 多周期控制器FSM

每个状态一个时钟周期。

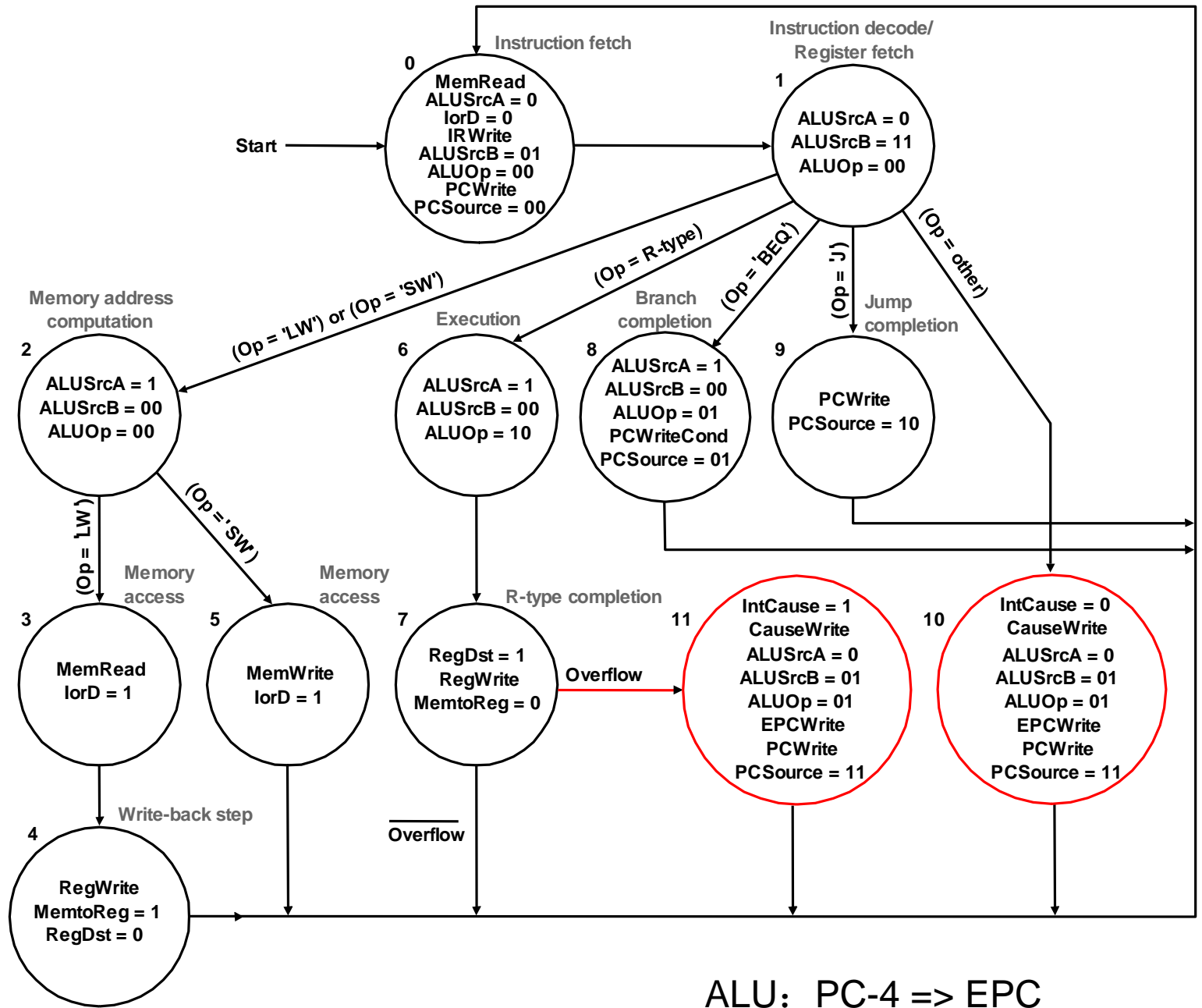


指令异常:  
非法指令  
算术溢出  
缺页



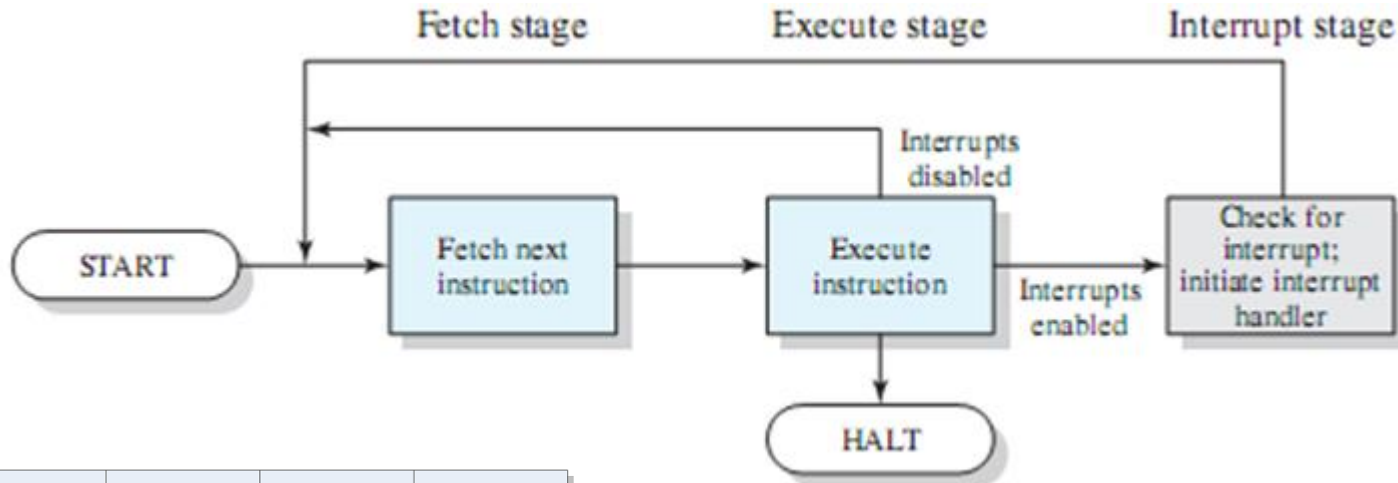
|    |    |    |     |    |
|----|----|----|-----|----|
| IF | ID | EX | MEM | WB |
|----|----|----|-----|----|

# 异常处理控制

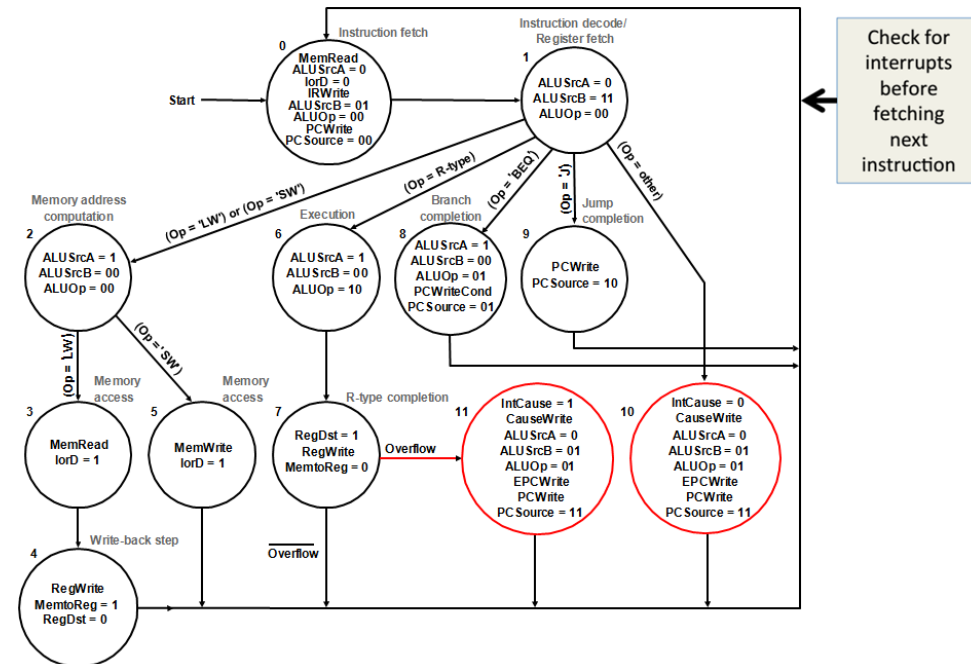


ALU: PC-4 => EPC

# Multicycle Interrupts: 中断周期

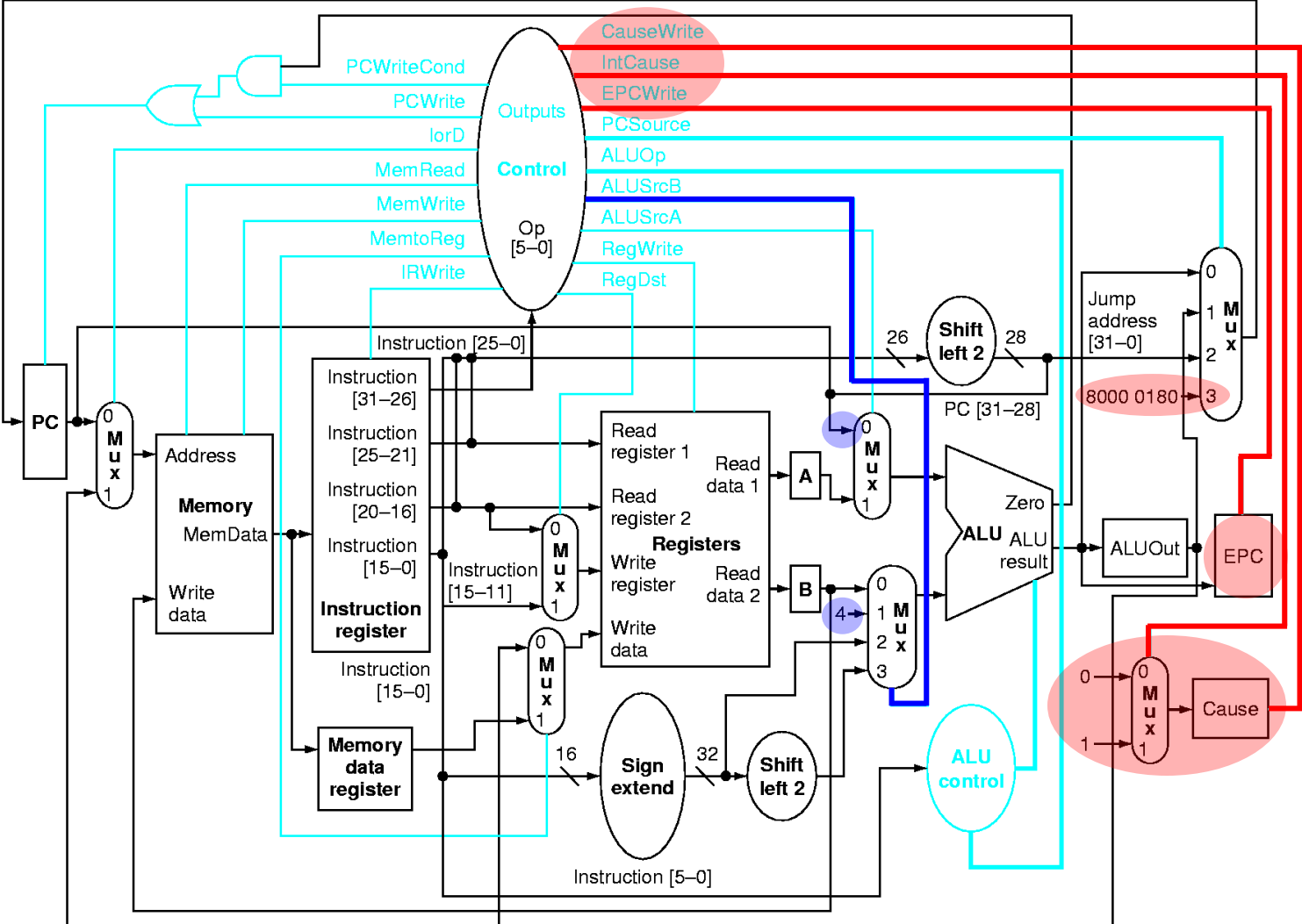


- CPU
  - Enable
  - EPC, Cause
- IntrCtrl:
  - Mask, 判优

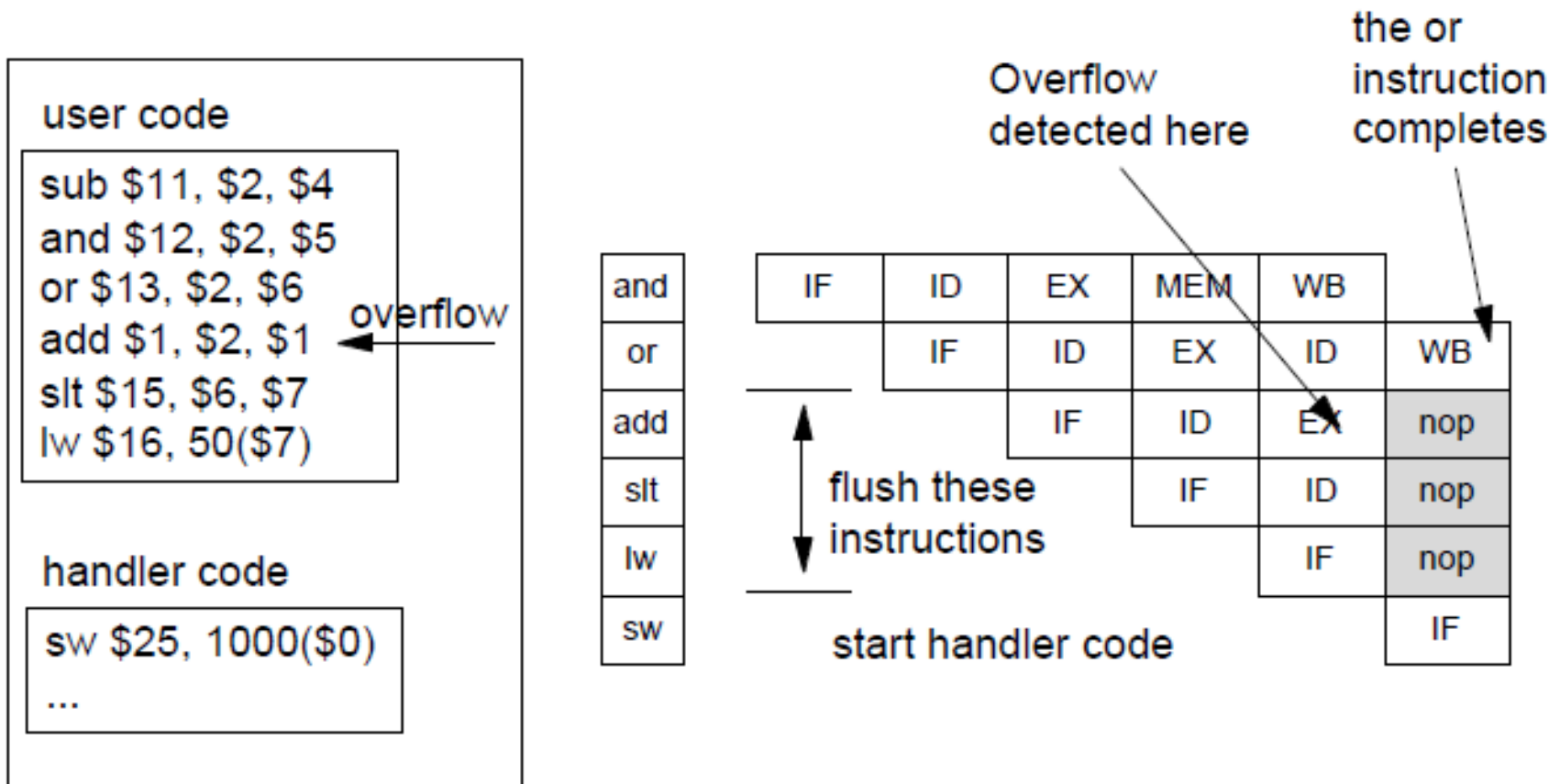




# Exceptions Handling in Multi-Cycle MIPS



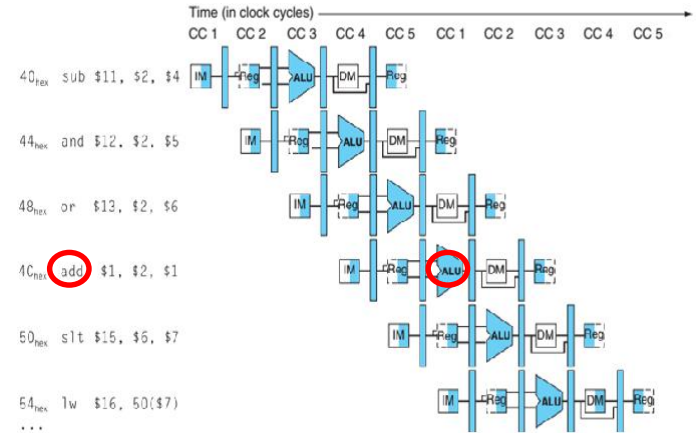
## \$4.9.2: add指令溢出异常的响应



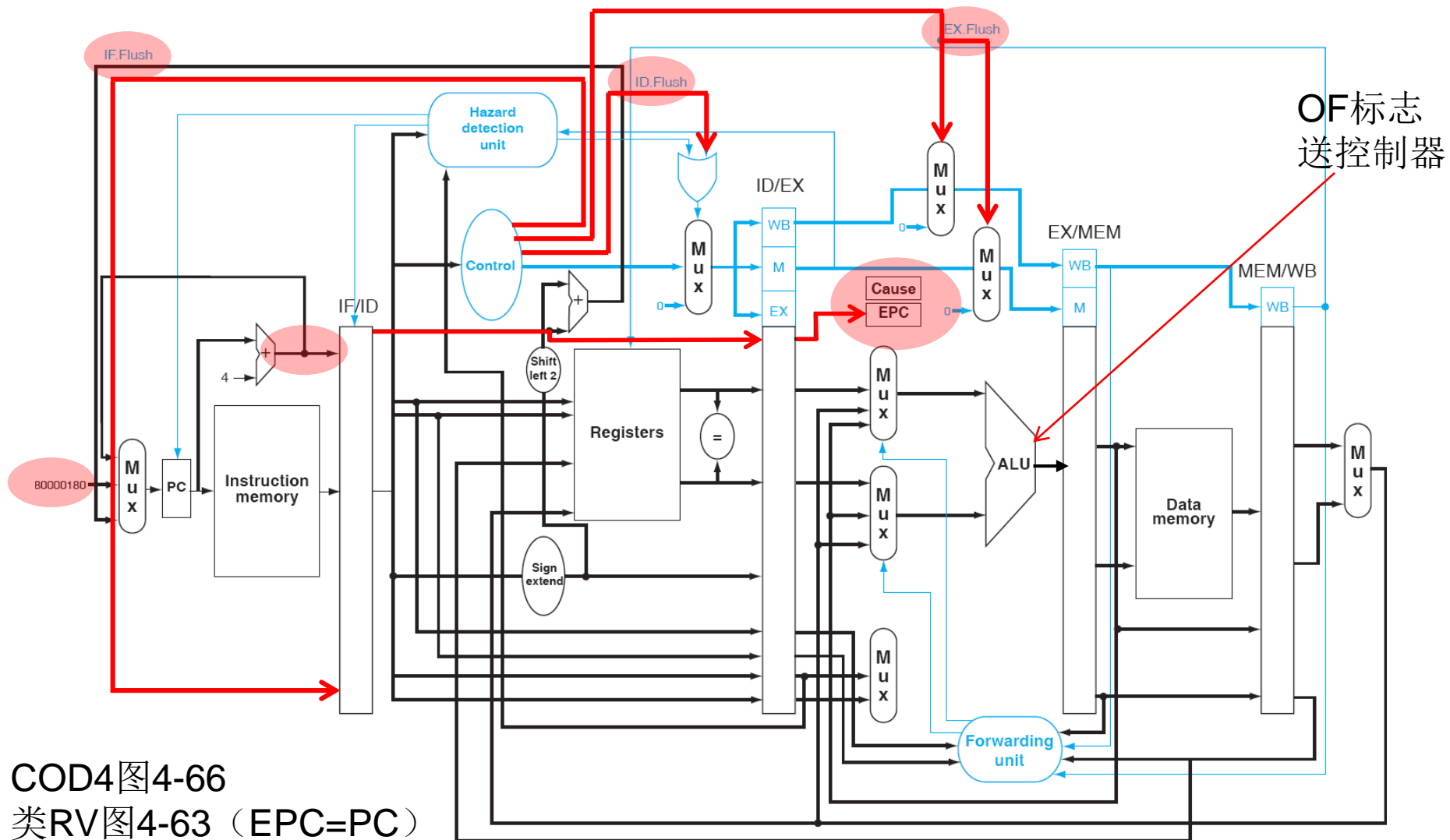
- 保证异常响应的顺序与指令执行的逻辑顺序相同  
– 之前的指令完成，之后的指令取消

# \$4.9.2: RV的add指令溢出异常

- 将异常视为一种**控制相关**
  - 执行完之前的所有指令
  - **flush**异常指令及之后的所有指令
    - add: IF.Flush, ID.Flush, **EX.Flush**
  - 记录异常原因: Cause
  - 保存断点: **EPC = 异常指令PC**, 精确!
  - 转异常处理程序: 非向量中断 (RV tvec@CSR)
  - 异常返回: **EPC**, 重新执行异常指令
    - “溢出”一般放弃, 其他(缺页)可能继续
- 遵从“简约原则”
  - 硬件仅提供最少支持, 其他由OS负责 (如分析Cause)



# add指令 overflow: flush IF/ID/EX



COD4图4-66

类RV图4-63 (EPC=PC)

第一个周期：检测，记录 (EPC、CAUSE)，排空，ESR入口写PC

第二个周期：取ESR的第一条指令

# RV图4-64, CC6

Overflow detected  
deasserting add

```

40hex sub x11, x2, x4
44hex and x12, x2, x5
48hex or x13, x2, x6
4Chex add x1, x2, x1
50hex sub x15, x6, x7
54hex ld x16, 100(x7)
. . .

```

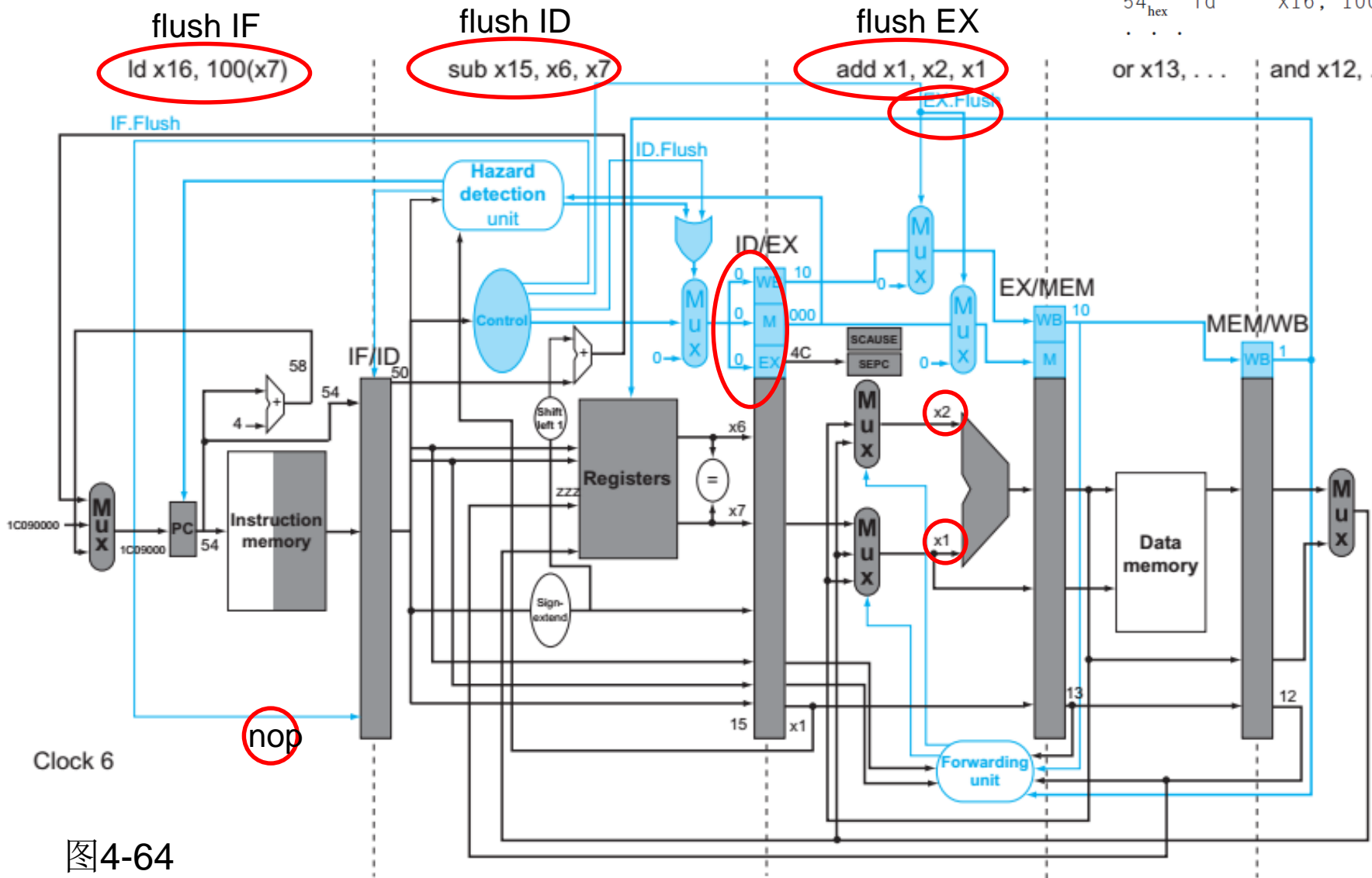


图4-64

# RV图4-64, CC7

```

40hex sub    x11, x2, x4
44hex and    x12, x2, x5
48hex or     x13, x2, x6
4Chex add   x1,  x2, x1
50hex sub    x15, x6, x7
54hex ld     x16, 100(x7)
. . .

```

first instruction of exception routine

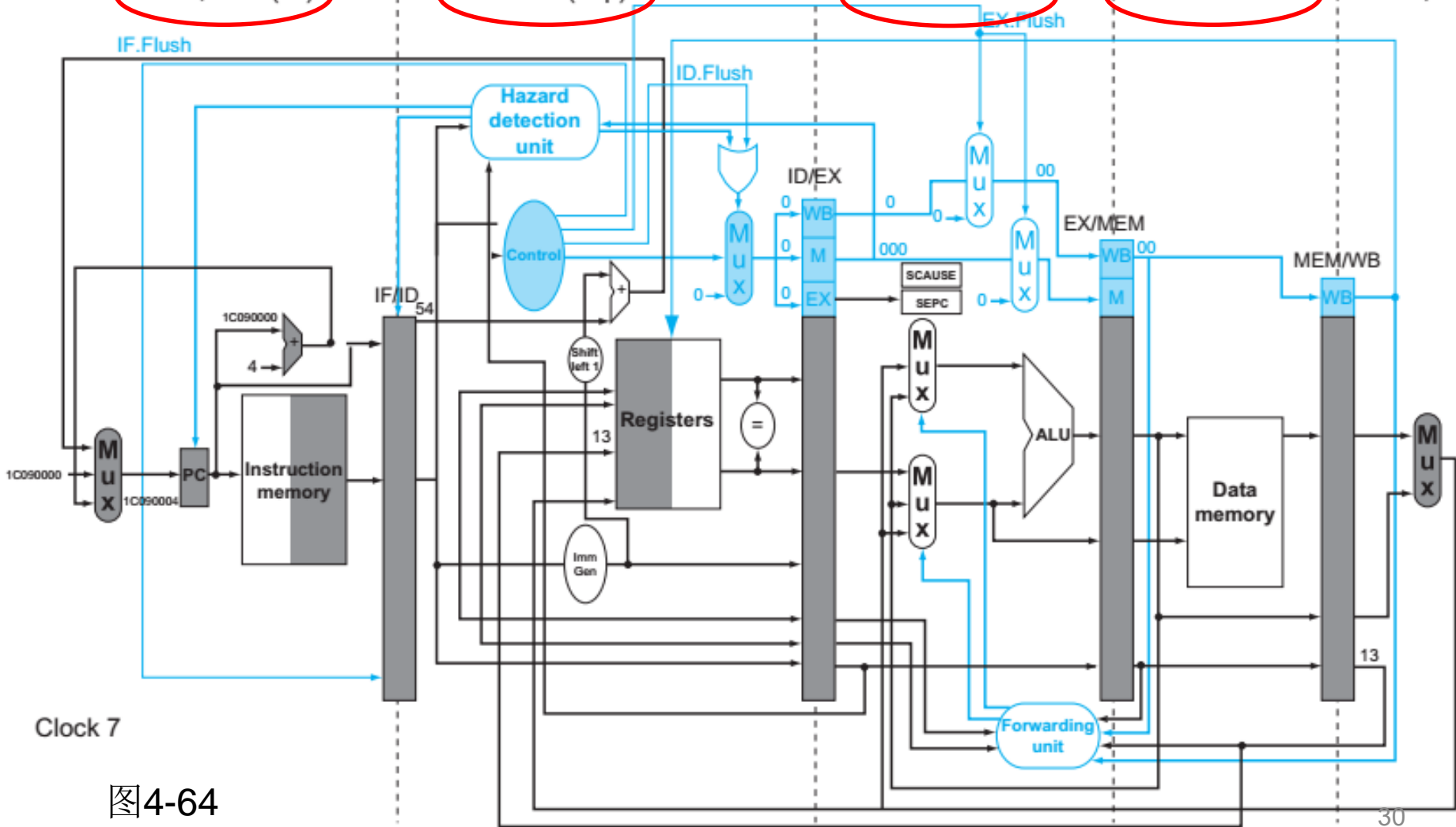
**sd x26, 1000(x0)**

**bubble (nop)**

**bubble**

**bubble**

or x13, ...

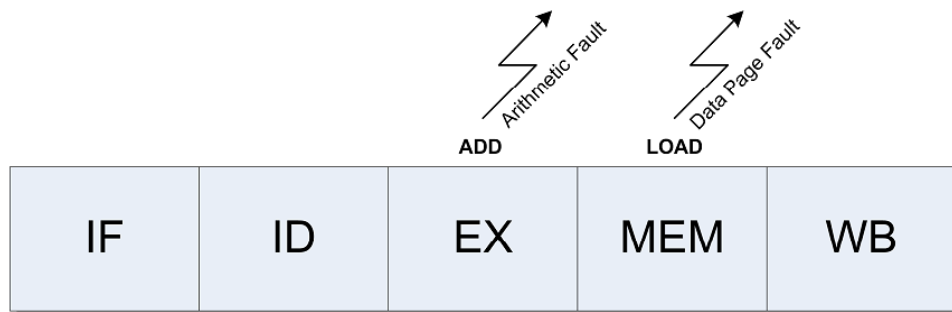
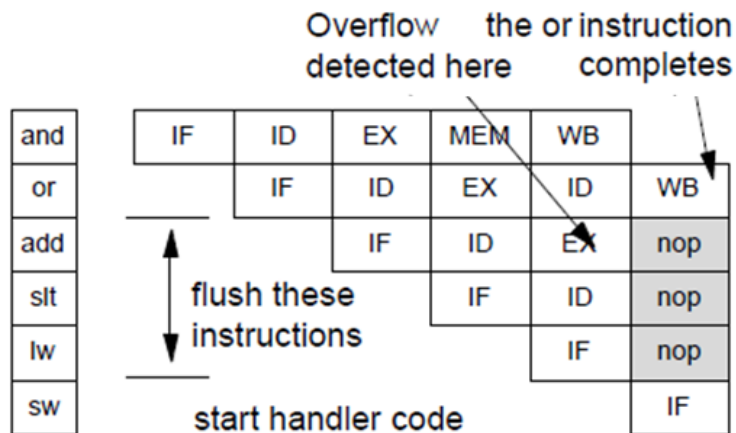


Clock 7

图4-64

# 上例的异常处理的问题： 仅单指令EX段异常！

- 单指令多异常： FCFS
  - IF段： 缺页
  - ID段： 非法指令
  - EX段： 溢出
  - MEM段： 缺页
  - WB段： 无（写寄存器错？）
- 多指令同一周期异常： FCFS
  - 精确： EPC=异常指令，如add
  - 非精确： EPC=IF段指令，如lw
    - 处理复杂： 需要OS依据流水线深度确定EPC
  - 在异常响应前，已经改变了系统的部分状态？
- 多中断： 优先级
  - 嵌套
- 异常+中断
- 乱序执行



# 精确异常实现：提交点技术？

- 提交点：MEM段（可否其他段？）

- 指令异常：发生最早者

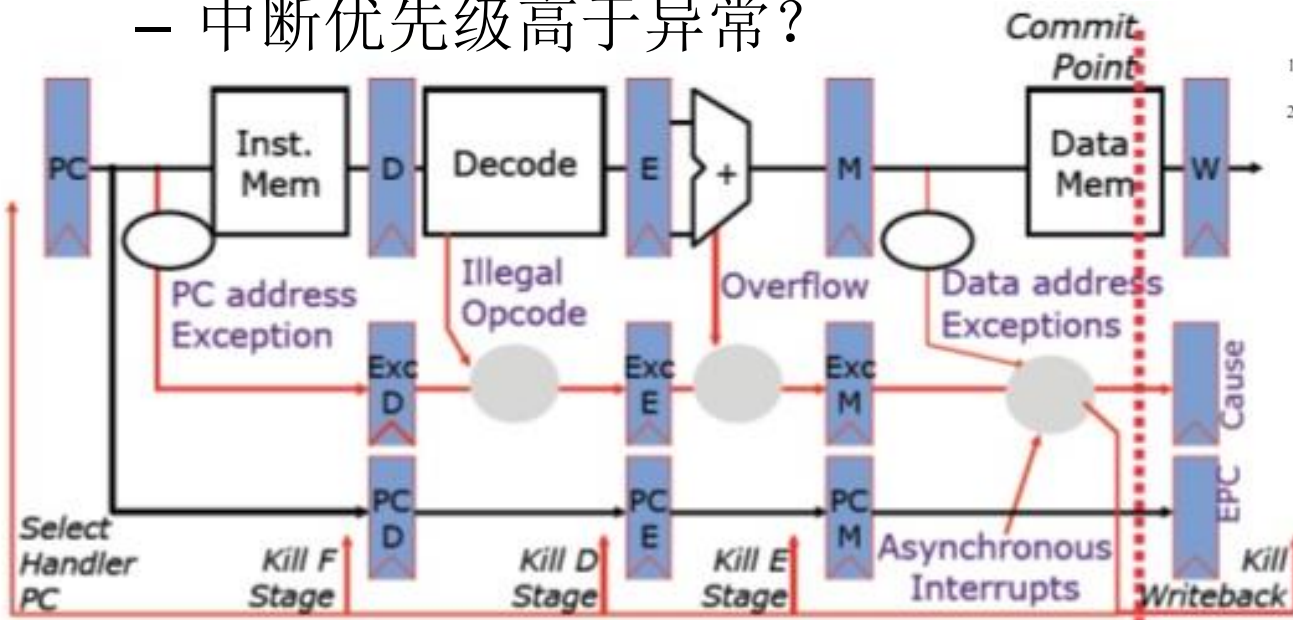
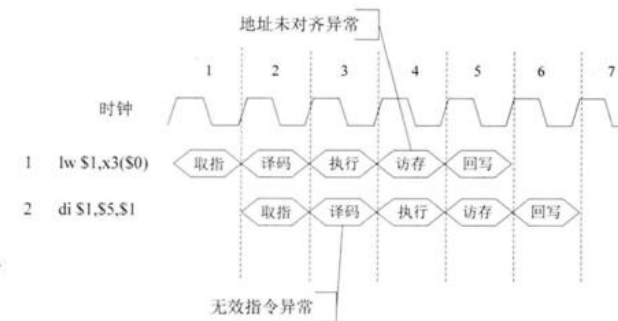
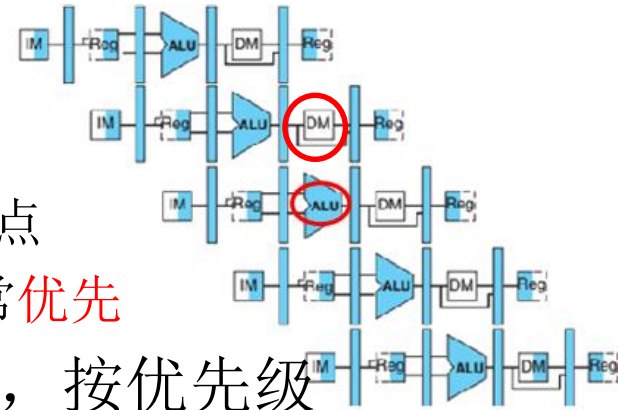
- 标记+气泡：异常并不立即响应，直至提交点

- 多指令：流水线中最深（最早）的指令异常优先

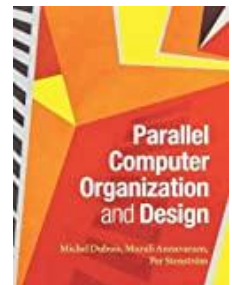
- 中断：M段检测（符合“异步”语义），按优先级

- 中断EPC = 当前MEM段指令

- 中断优先级高于异常？

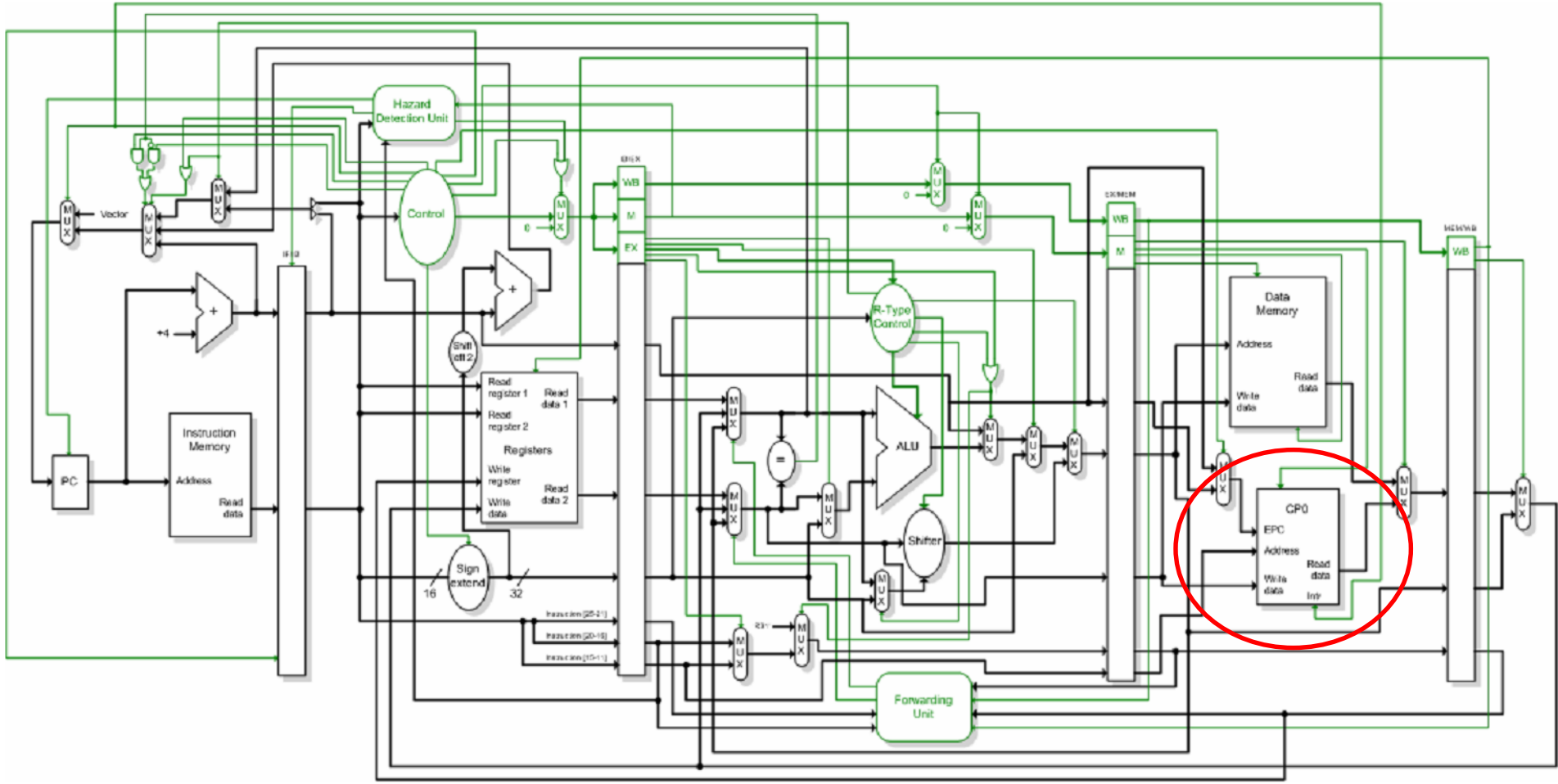


\$3.3.1

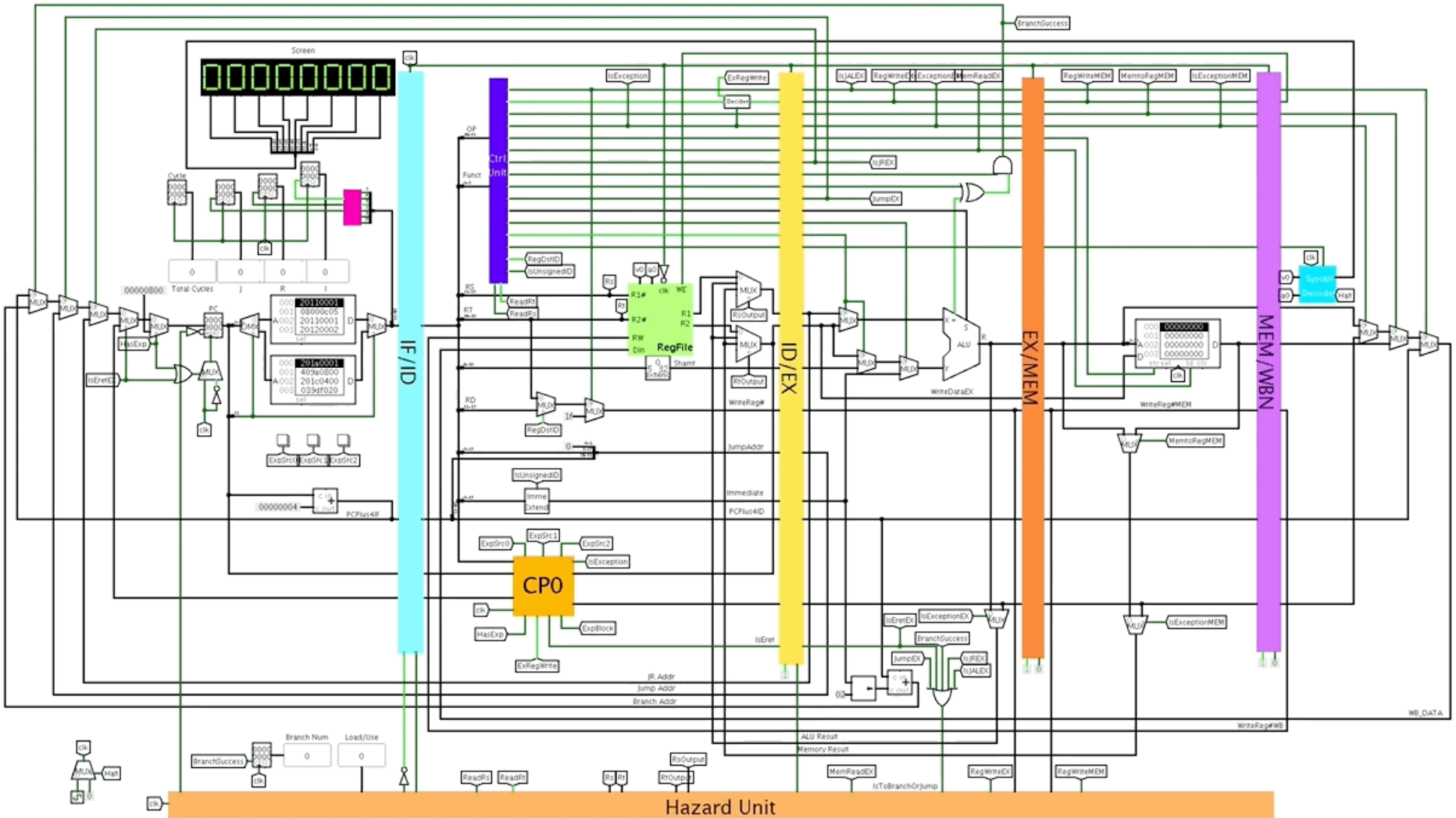




# PH processor: Pont@Univ of Leicester

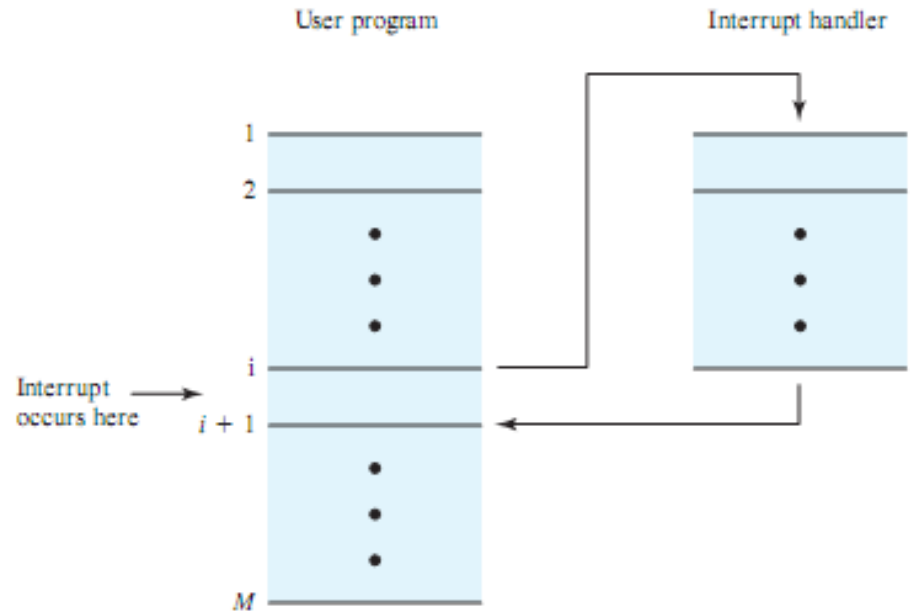


# 仿真器☺



# 小结

- 异常，中断
  - 断点
- 响应过程
  - 入口地址
  - 现场保存与恢复
- 过程控制
  - 多周期
  - 流水线：分支转移



# 思考，作业

- 中断周期要完成哪些微操作？
- 多周期状态机中，出现溢出的指令是否将错误结果写回？
- 多周期中状态机中，如何响应中断？
- 指令顺序执行，中断“精确”；指令流水执行，中断“精确”或“非精确”可选？
- 流水线是否存在“中断周期”？
- 为何提交点是M段？
- EPC和cause应该在哪个段？异常检测电路？
- RV异常返回指令eret如何实现？
- 异常与中断同时发生，优先级？
- 比较中断、异常、陷阱、过程调用
  - 请求时间、响应时间，断点与现场，返回点，同步异步，中断周期、系统状态，参数传递，控制转移？
- 作业：4.30

