Module 3: Processor Design
Exceptions

Module 3.4 Exceptions

Agenda

1. Types of exceptions
2. Exception handling in Multicyle MIPS
3. Exception handling in Pipelined MIPS

- **References**
  - **Patterson Hennessy Section 4.9**
  - **Hennessy Patterson Apendix C.4**

1. Types of Exceptions

Two types

- **Interrupts**
  - Produced by external events (external signal activated by a peripheral)
  - They are asynchronous to the program execution
  - Not related to any phase of any instruction
- **Exceptions**
  - Internal events that trigger an exception
    - Overflow, internal errors, page misses, …
  - Synchronous with the program
    - Associated to one phase of the execution of one instruction
  - The program must stop the regular execution just at that instruction and resume on the same or the next (depends on the exception) after the exception has been serviced

2. Exceptions on Multicycle MIPS

We focus on two exceptions: **Undefined instruction (ID) and Overflow (Ex)**

Basic actions:

- **Save the PC** of the interrupted instruction **in a *EPC register*** (exception program counter)
- Transfer control to the O.S. by writing on the PC a specific address
- **The O.S. sets on that addres a branch to a ISR**, which handle the exception
- **The return from the ISR will restore the context,** copying the content of the EPC into the PC (the execution will resume where it was before the exception)

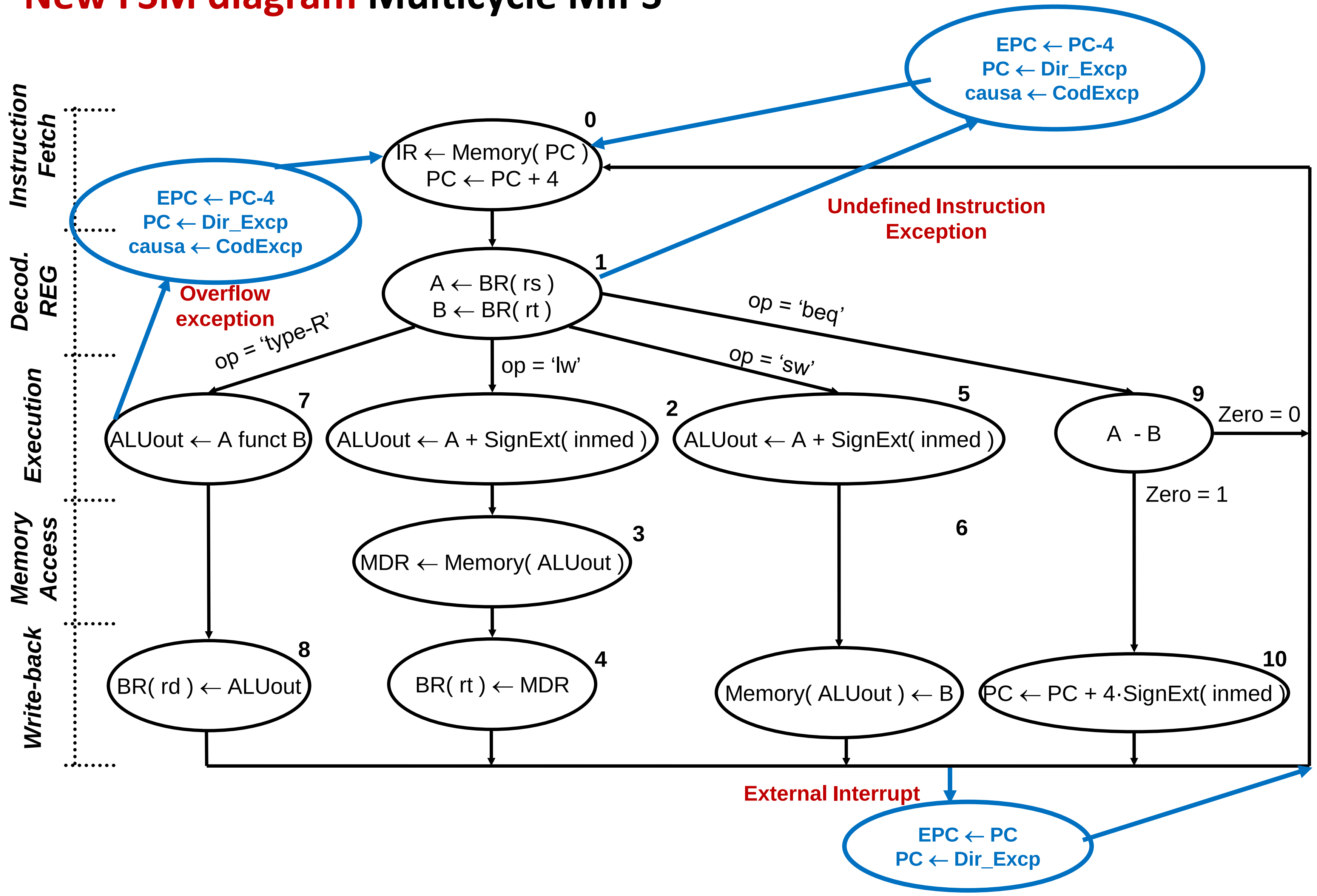
2. Exceptions on Multicycle MIPS

Hardware added to manage the execptions:

- **EPC Register** (exception program counter)
- **Cause Register**: indicate the casuse of the exception:
  - Bit 0: Undefined instruction
  - Bit 1: Arithmetic Overflow
- **Control signals**:
  - **EPCwrite** → Write enable for EPC (EPC <= PC – 4)
  - **CauseWrite** → Write enable for Cause
  - **IntCause** → Data written on Cause
- One input to the mux connected to the Address bus of the memory, to select the address associated to the exception (exception vector)

2. Exceptions on Multicycle MIPS

New FSM diagram Multicycle MIPS



3. Exceptions on the Pipelined MIPS

Any instruction in the pipeline can generate an exception

IF	Page Miss; Unaligned Access; Protection fault
ID	Undefined instruction
Ex	Arithmetic overflow
MEM	Page Miss; Unaligned Access; Protection Fault
WB	None

– **Problems:**
- When should an exception be handled?
- If several exceptions happen, which should we attend first?
- Can we safely change the processor status to manage the exception?

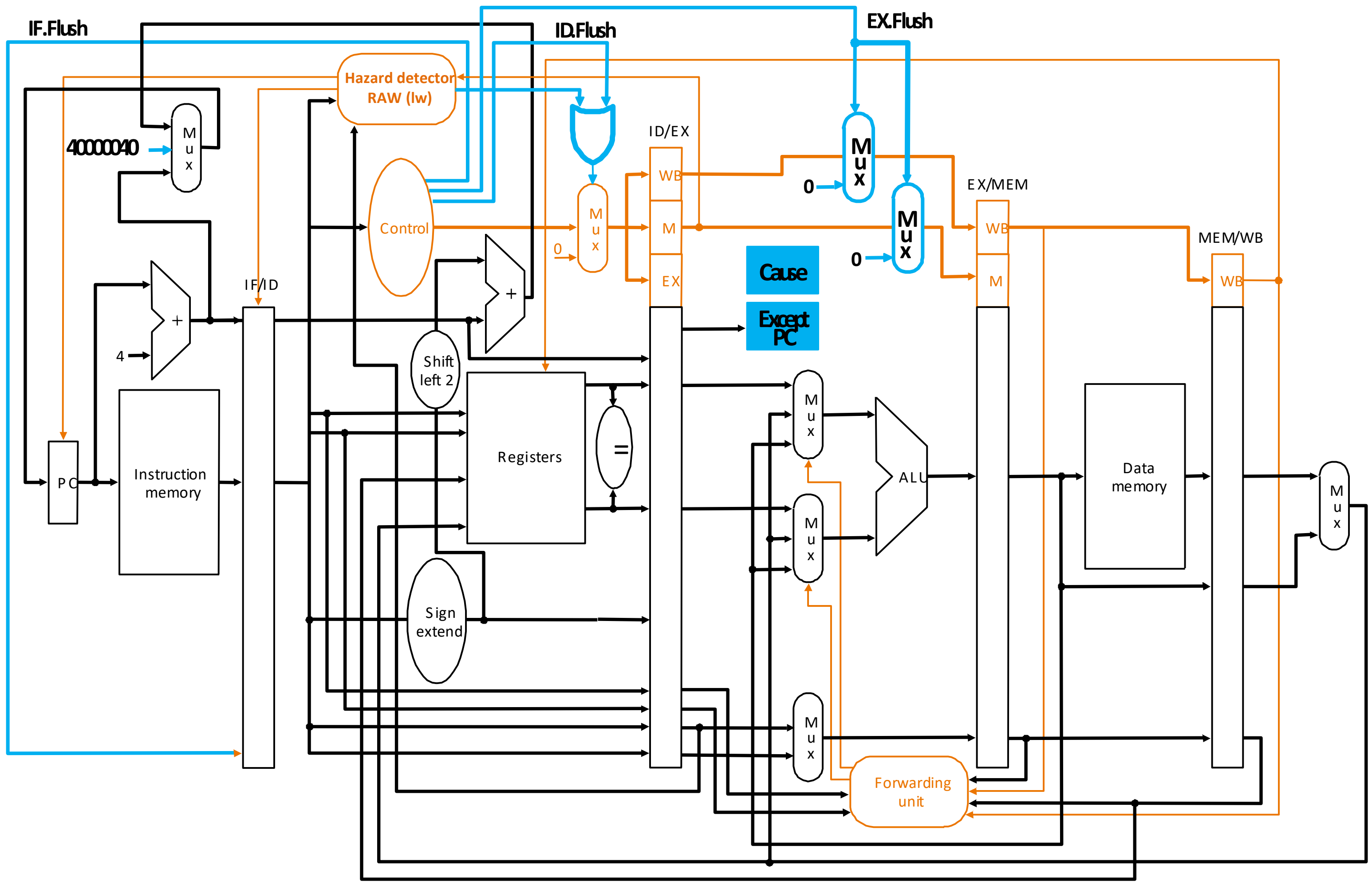
3. Exceptions in Pipelined MIPS

- Precise exception handling:
- When an exception happens
  - All instructions that entered the pipeline before the instruction that triggers the exception must complete their execution
  - The instructions that entered the pipeline after the instruction that triggers the interrupt must be canceled, and its execution must be resumed after the ISR returns as if they had not entered the pipeline before
- The cpu must recover the status it had before the exception handling started
- **Some exceptions imply the re-execution of the instruction that triggered the exception**:
  - **E.g. Page miss**
- External Interrupts (I/O)
  - **Not associated to an instruction**
  - **Pipeline is flushed** (introducing NOPs)

3. Exceptions in the Pipelined MIPS

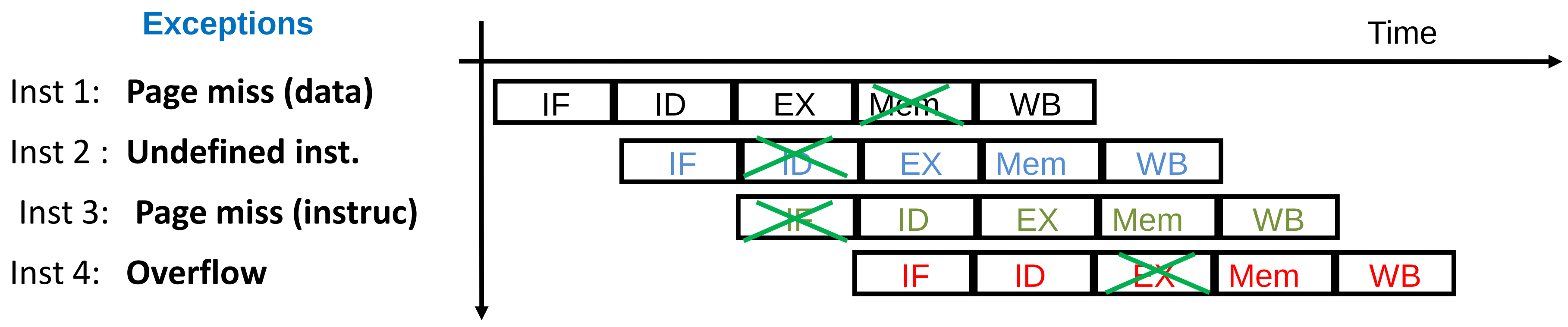
Hardware added to manage the exceptions

- **Registers "Cause" y EPC**; constant branch target address for the exception service routine **"4000040"**
- Generation of control signals **IF.Flush, ID.Flush, EX.Flush to flush the *pipeline***



3. Exceptions in the Pipelined MIPS

Let us assume the following situation:



– The exceptions should be handled in order

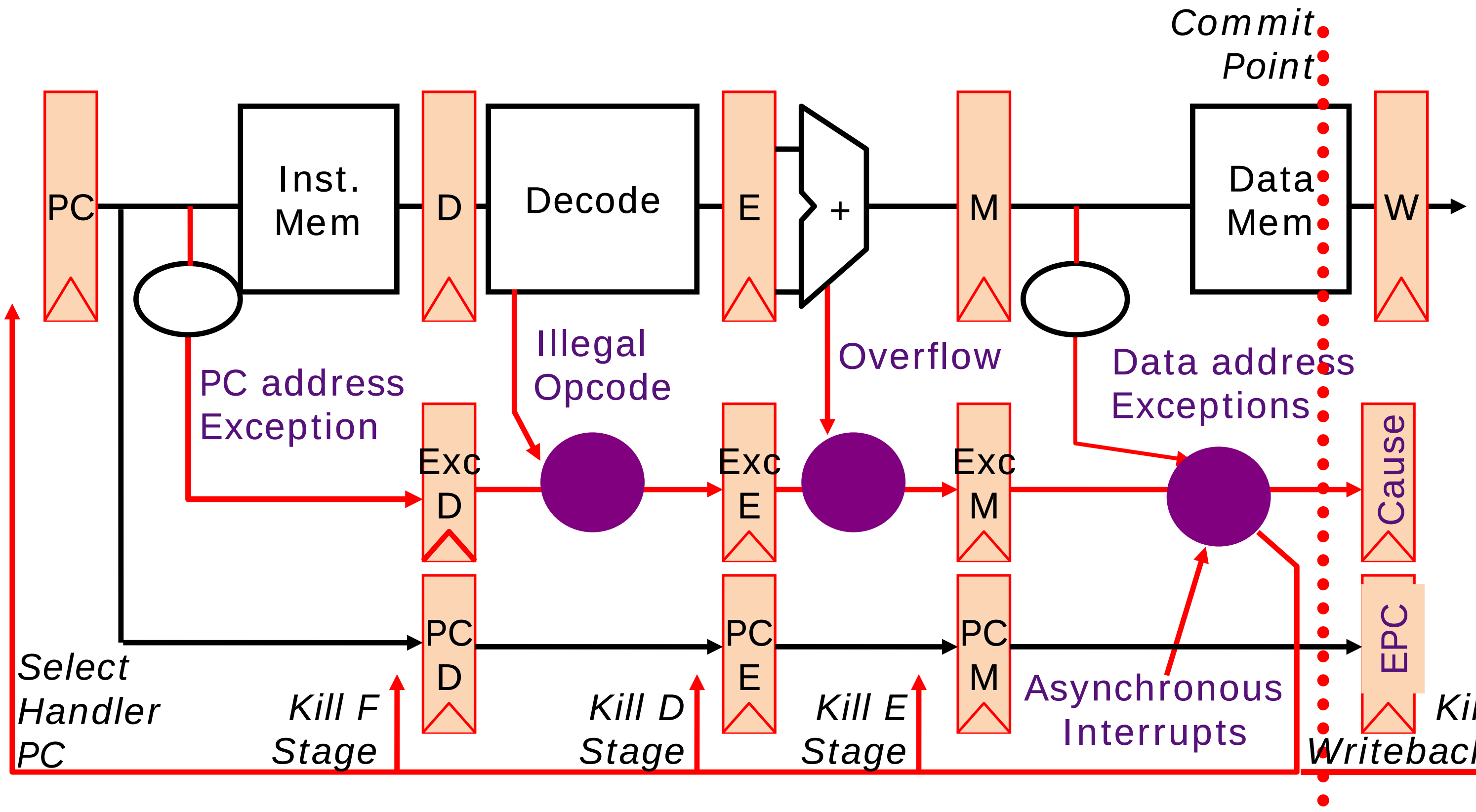
- In this example, the page miss in the data memory should by handled first

– **Problem**: the undefined instruction and page miss in instruction memory happen first (previous cycle)

3. Exceptions in the Pipelined MIPS

Solution

- Add a exception register and pc copy to the pipeline registers
- The exception register has one bit for each stage/exception source
- Each stage sets the corresponding bit if an exception happens that stage
- When a exception bit is activated the wirte enable signals are deactivated for that instruction
- The exception bits are checked in the WB stage


Commit Point
PC
Inst. Mem
D
Decode
E
+
M
Data Mem
W
PC address Exception
Illegal Opcode
Overflow
Data address Exceptions
Exc D
Exc E
Exc M
Cause
PC D
PC E
PC M
EPC
Asynchronous Interrupts
Select Handler PC
Kill F Stage
Kill D Stage
Kill E Stage
Kill Writeback