COMPUTER ORGANIZATION AND DESIGN

Mahesh Awati
Department of Electronics and Communication Engineering

The Processor

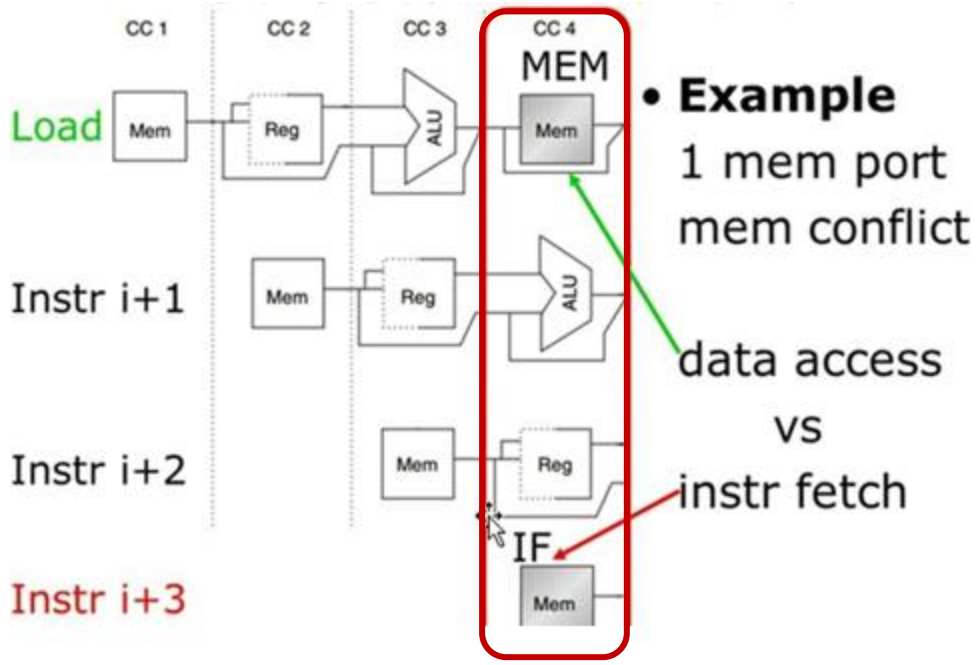
Structural Hazards

Structural Hazard

- It means that the **hardware cannot support the combination of instructions that we want to execute in the same clock cycle.**
- When a planned instruction cannot execute in the proper clock cycle because the hardware does not support the combination of instructions that are set to execute

Assume a single memory instead of two memories (Single Memory holding Code and Data) -

- If the 5-stage pipeline had a fourth instruction, **we would see that in the same clock cycle, the first instruction is accessing data from memory while the fourth instruction is fetching an instruction from that same memory**. Without two memories, our pipeline could have a structural hazard



COMPUTER ORGANIZATION AND DESIGN

4.9 Control Hazards

Mahesh Awati
Department of Electronics and Communication Engineering

The Processor

Control Hazards/ Branch Hazard: Stalls

- **Control Hazards arise** when there is **need to make decision based on results of an Instruction** while others are executing.
- **Example:** A conditional Branch Instruction(beq)
- A conditional Branch Instruction (beq) has to
  a) **decide on New value for PC** i.e.,
  PCnew= PCpre+4 or
  PCnew=PCpre + Sign Extended offset<<1
  based on **ZERO output**
- In this, **branch outcome determined in MEM** as per the datapath.

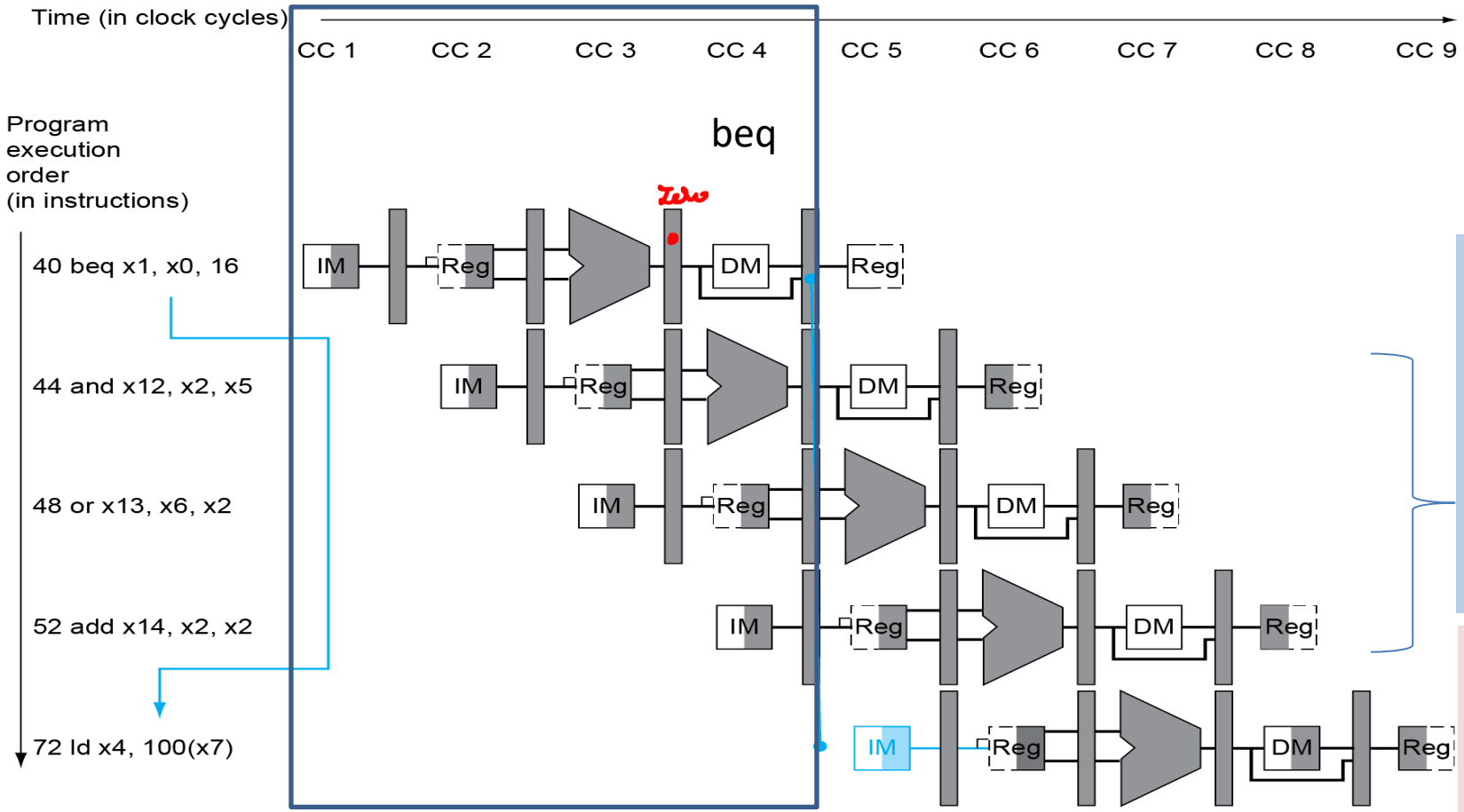
Let us consider the set of following Instructions and check if there is any hazard when they go through pipeline??

```
beq x1,x0,16
and x12,x2,x5
or x13,x6,x2
and x14,x2,x2 // PC-relative branch to 40+16*2=72
lw x4,100(x7)
```

The Processor

Control Hazards/ Branch Hazard: Stalls

Branch outcome determined in MEM



In cc4, beq is in MEM stage zero status is know. **Based on zero, It will decide on branching**. But, we can see that **already next 3 instructions are in pipeline. If the zero =1 , then these instructions need not to be executed.**

- ✓ Flush these instructions (Set control values to 0).
- ✓ Execution getting delayed **by 3 clocks**

The Processor

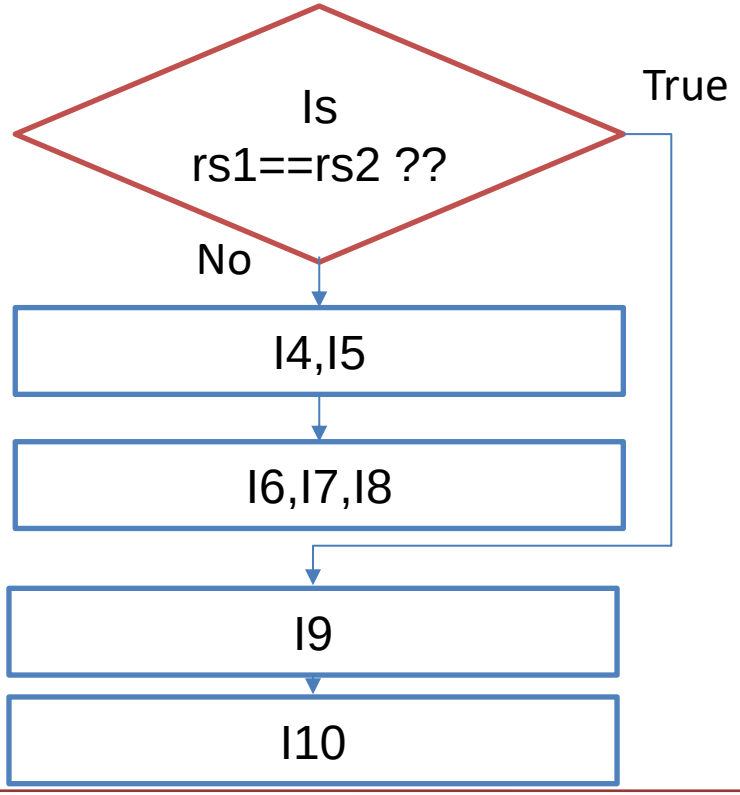
Control Hazards/ Branch Hazard: Stalls

Performance via Prediction

In some cases, **it can be faster on average to guess and start working rather than wait until you know for sure**, assuming that the mechanism to recover from a mis-prediction is not too expensive and your prediction is relatively accurate.

PC →

Address	Label	Instructions
0x1000		I1
0x1004		I2
0x1008		bneq Next
0x100C		I4
0x1010		I5
0x1014		I6
0x1018		I7
0x101C		I8
0x1020	Next:	I9…….



The Processor

Control Hazards/ Branch Hazard: Stalls

Execution of branch instruction or branching by the direct modification of PC.

By the time a branch instruction goes through the execution stage of the pipeline and reaches the MEM stage, regardless of the branch instruction outcome, It has fetched the next 3 successive instructions and are EX,ID and IF stages in the pipeline. Therefore the RISC-V core will **flush its pipeline**.

Cycle	IF	ID	IE	MA	WB
1	I1	X	X	X	X
2	I2	I1	X	X	X
3	I3	I2	I1	X	X
4	I4	I3	I2	I1	X
5	I5	I4	I3	I2	I1
6	I6	I5	I4	I3	

if I3 is a Conditional Branch Instruction

If branch condition is TRUE Branch location gets updated in PC in by PCSrc – which is generated in MEM stage in RISC-V

PC →

Address	Label	Instructions
0x1000		I1
0x1004		I2
0x1008		BZ Next
0x100C		I4
0x1010		I5
0x1014		I6
0x1018		I7
0x101C		I8
0x1020	Next:	I9…….

The Processor

Control Hazards/ Branch Hazard: Stalls

Execution of branch instruction or branching by the direct modification of PC.

By the time a branch instruction gets into execution stage of pipeline, regardless of branch instruction outcome, It has fetched next successive instruction and are in pipeline. Therefore the RISC-V core will **flush its pipeline**.

Cycle	IF	ID	IE	MA	WB
1	I1	X	X	X	X
2	I2	I1	X	X	X
3	I3	I2	I1	X	X
4	I4	I3	I2	I1	X
5	I5	I4	I3	I2	I1
6	I6	I5	I4	I3	I2
7					
8					

if I3 is a Conditional Branch Instruction

PC

0x1020	Next:	I9

If branch condition is TRUE Branch location gets updated in PC in by PCSrc – which is generated in MEM stage in RISC-V

FLUSH the Pipeline

What will happen to **I4 and I5 which are in Pipeline ????**

The Processor

Control Hazards/ Branch Hazard: Stalls

Execution of branch instruction or branching by the direct modification of PC.

By the time a branch instruction gets into execution stage of pipeline, regardless of branch instruction outcome, It has fetched next successive instruction and are in pipeline. Therefore the RISC-V core will **flush its pipeline**.

Cycle	IF	ID	IE	MA	WB
1	I1	X	X	X	X
2	I2	I1	X	X	X
3	I3	I2	I1	X	X
4	I4	I3	I2	I1	X
5			I3	I2	I1
6				I3	I2
7	I9	X	X	X	I3
8	I10	I9	X	X	X

I4, I5, and I6 which were in the pipeline will be FLUSHED & Refilling of the pipeline starts from the branched location

PC →

Address	Label	Instructions
0x1000		I1
0x1004		I2
0x1008		BZ Next
0x100C		I4
0x1010		I5
0x1014		I6
0x1018		I7
0x101C		I8
0x1020	Next:	I9…….

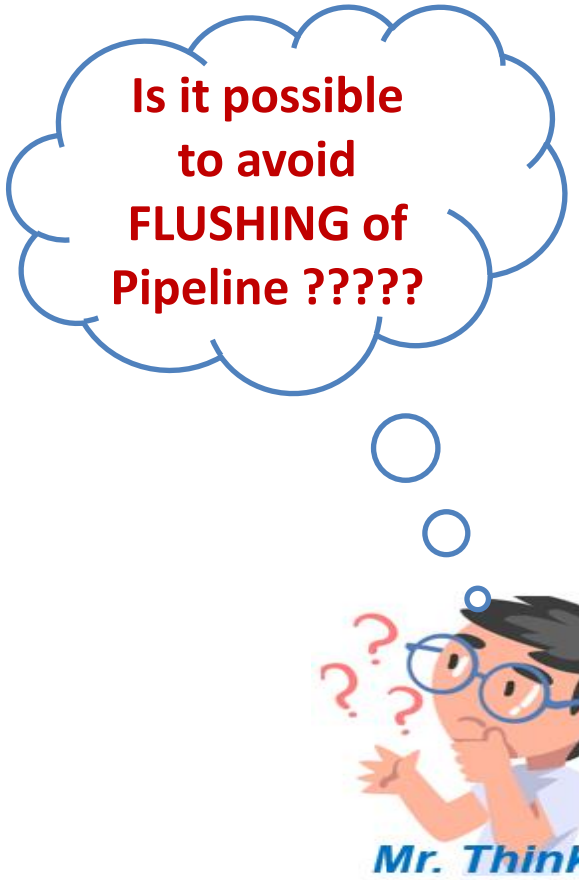
The Processor

Control Hazards/ Branch Hazard: Stalls

Branch Prediction

In computer architecture, a **branch predictor** is a digital circuit that tries to guess **which way** a **branch** will go before this is known definitively. This reduces effect of pipeline flush.
Branch prediction reduces the effect of a pipeline flush by predicting possible braches and loads the new branch address prior to the execution of the instruction

One approach is to look up the address of the instruction to see if the conditional branch was taken the last time (History) this instruction was executed, and, if so, to begin fetching new instructions from the same place as the last time. This technique is called dynamic branch prediction.



The Processor

Control Hazards/ Branch Hazard: Stalls

Hardware Solution for Reducing Branch Delay due to Flushing : Can the branch delay because of FLUSHing be reduced below 3 clocks ?

Move **hardware to determine outcome** of Branch **from MEM stage to ID stage**

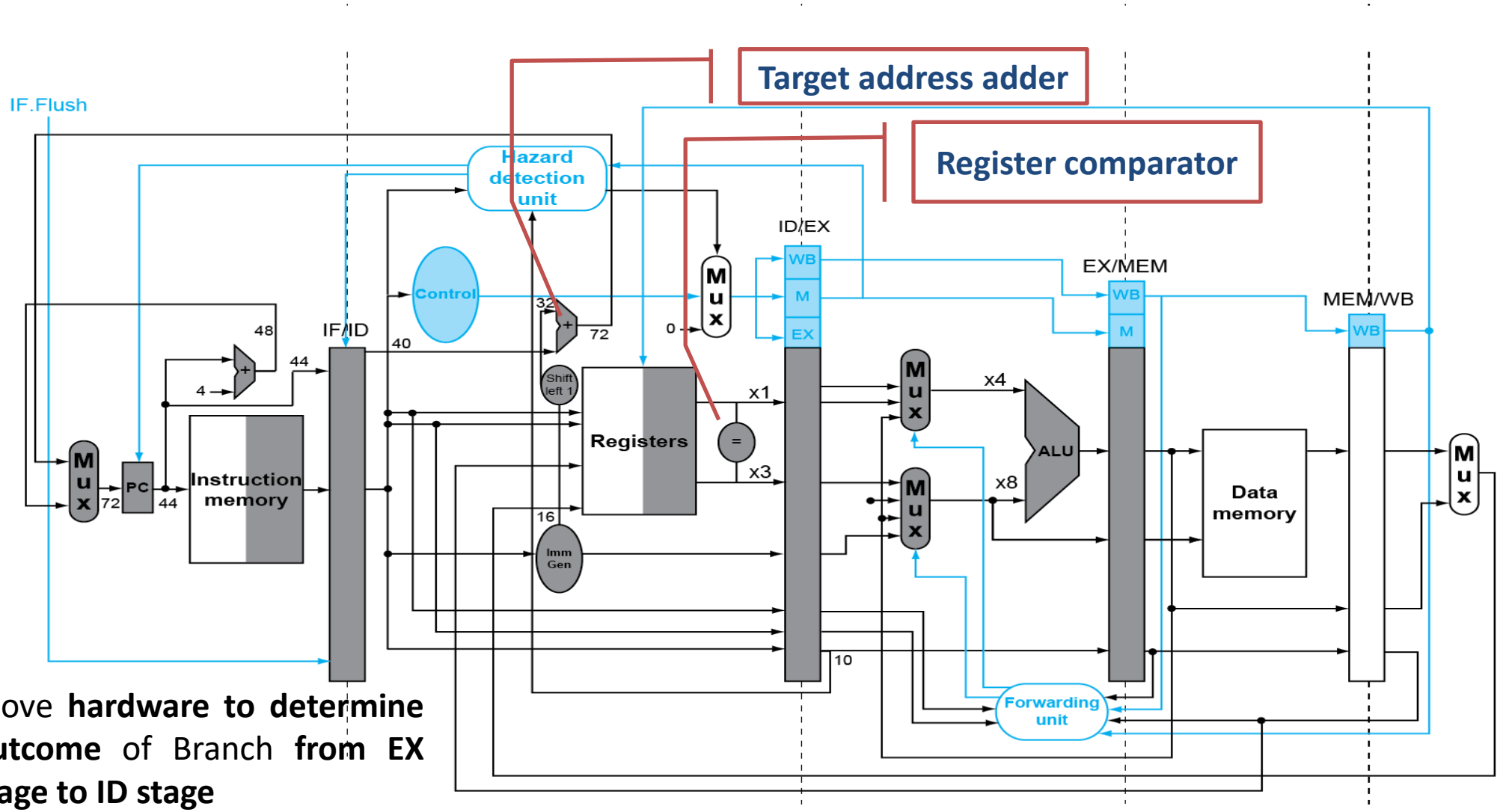
a) **Target address adder : Used to compute PCnew=PCpre+offset<<1**
b) **Register comparator : Comparison of rs1,rs2**

Example: branch taken

```
36:   sub   x10, x4, x8
40:   beq   x1,  x3, 16   // PC-relative branch to 40+16*2=72
44:   and   x12, x2, x5
48:   orr   x13, x2, x6
52:   add   x14, x4, x2
56:   sub   x15, x6, x7
      ...
72:   ld    x4, 50(x7)
```

The Processor

Control Hazards/ Branch Hazard: Stalls

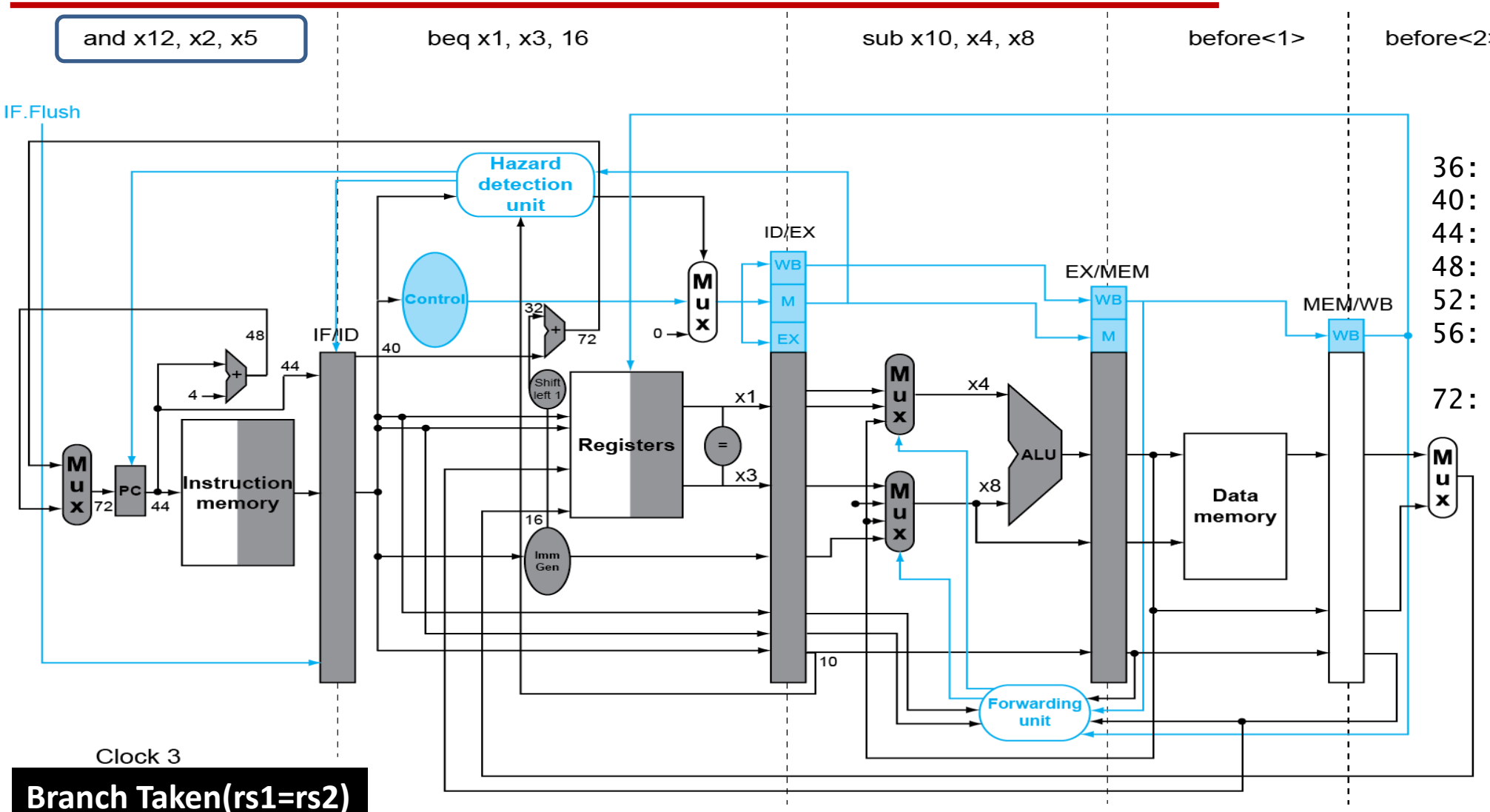


Move **hardware to determine outcome** of Branch **from EX stage to ID stage**

The Processor

Control Hazards/ Branch Hazard: Stalls

Effect of Moving **hardware to determine outcome** of Branch **from EX stage to ID stage**




```
36:   sub   x10, x4, x8
40:   beq   x1,  x3, 16
44:   and   x12, x2, x5
48:   orr   x13, x2, x6
52:   add   x14, x4, x2
56:   sub   x15, x6, x7
          ...
72:   ld    x4, 50(x7)
```

Observation:
and x12,x2,x5 is already there in pipeline

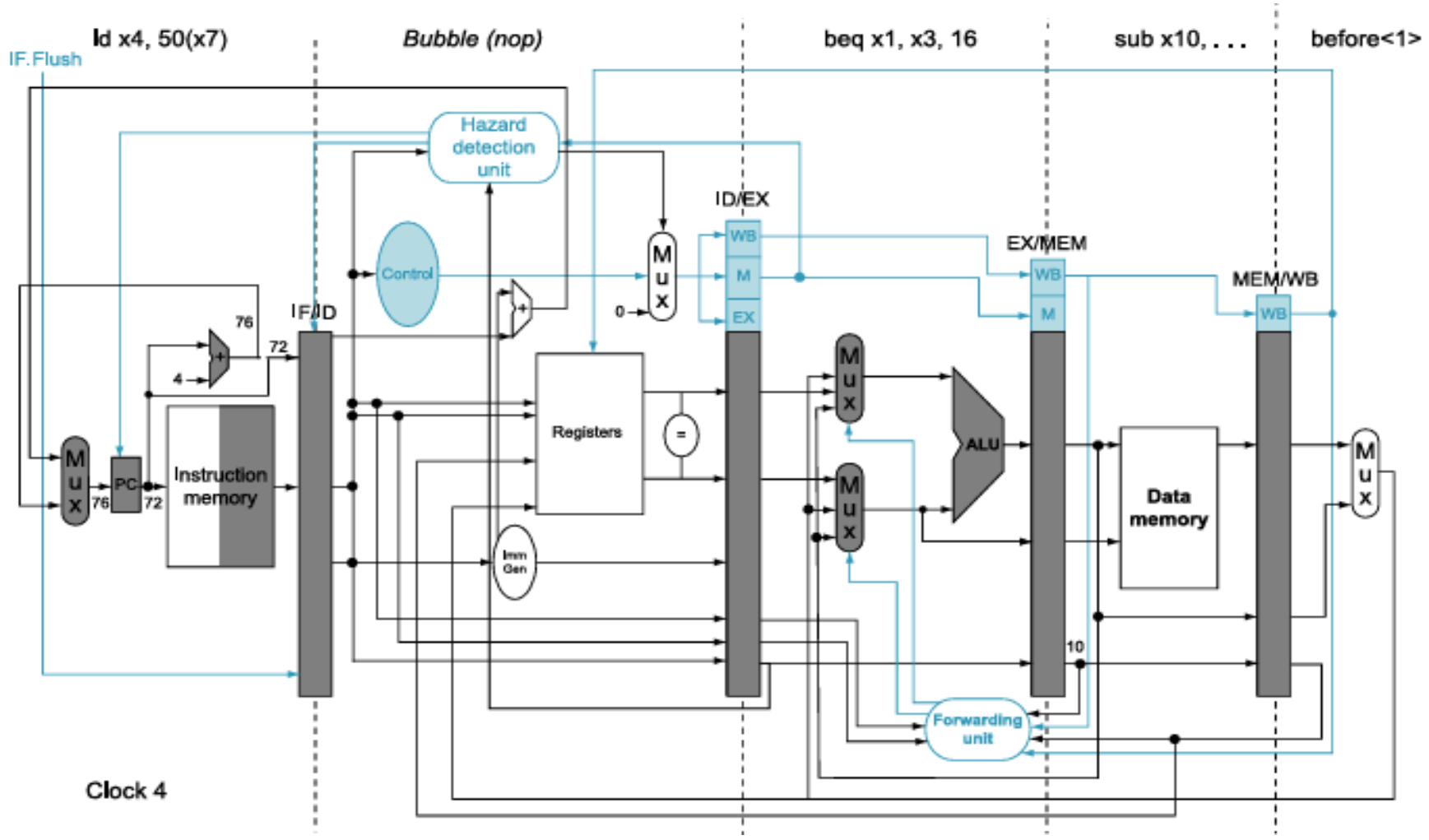
Clock 3

Branch Taken(rs1=rs2)

The Processor

Control Hazards/ Branch Hazard: Stalls

Effect of Moving **hardware to determine outcome** of Branch **from EX stage to ID stage**




```
5:   sub   x10, x4, x8
0:   beq   x1,  x3, 16
4:   and   x12, x2, x5
3:   orr   x13, x2, x6
2:   add   x14, x4, x2
5:   sub   x15, x6, x7
     ...
2:   ld    x4, 50(x7)
```

Observation:
and x12,x2,x5 is already there in pipeline

Branch Taken(rs1=rs2)

The Processor

Control Hazards/ Branch Hazard: Stalls

1. Assume a 5-stage pipelined architecture with Von Neuman bus architecture (common memory for code and data). What is the type of hazard it leads to?
   a. Data hazards
   b. Control hazards
   c. Structural hazards
   d. None
2. In the 5-stage pipelined architecture, the PC gets updated based on the outcome PCSrc control signal. Assume a branch Instruction is in the MEM stage and the outcome of the branch is TRUE. What type of hazard does it lead to & how many clocks would get wasted to resolve the hazard?
   a. Data Hazard, 3clk
   b. Structural Hazard, 3clk
   c. Control Hazard, 1clk
   d. Control Hazard, 3clk
3. Which are the techniques used to handle the control hazard in a 5-stage pipelined architecture?
   a. Branch predictor
   b. Flushing the instructions that have entered the pipeline after the branch instruction
   c. Flushing the instructions that have entered the pipeline before the branch instruction
   d. Both a & b
4. Assume the comparator & adder which were present in the execution stage are moved to the decoder stage. This modification leads to the outcome of the branch being known in the decoder stage, what will be the number of clks that will be wasted in flushing the instructions if the outcome of the branch is true?
   a. 3 clks
   b. 2 clks
   c. 1 clks
   d. 0 clks

The Processor

Control Hazards/ Branch Hazard: Stalls

1. Consider the following program, which is getting executed on a 5-stage pipelined architecture

Address	Instructions
36:	sub x10, x4, x8
40:	beq x1, x3, 16
44:	and x12, x2, x5
48:	or x13, x2, x6
52:	add x14, x4, x2
56:	sub x15, x6, x7
	...
	...
68	add x5, x5, x7
72:	ld x4, 50(x7)
74	

a) In the Program given, if the branch on equal is in the MEM access stage, then which instructions are in the WB, EX, ID, and IF stages?
b) If the outcome of the branch is true, what will the PC value be?
c) If the outcome of the branch is true which addressing mode is used to calculate PCnew?
d) If the outcome of the branch is true, which Instructions in the pipeline are to be flushed to handle control hazards, and how many clocks are getting wasted?

COMPUTER ORGANISATION AND DESIGN

4.10

Exceptions

Mahesh Awati
Department of Electronics and Communication Engineering

The Processor

Exceptions/Interrupts

Exceptions and **Interrupts**

- "Unexpected" events generated **Internally (within CPU) or externally (External I/O controller)** requesting for the service.
- **Exceptions are unusual conditions that occur at run time, associated with an instruction in the current RISC-V hart.**
- **Interrupts are events that occur asynchronously outside any of the RISC-V harts.**
- An exception/ interrupt will change the normal flow of instruction execution.
- Dealing with them without sacrificing performance is hard.

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

The Processor

Exceptions/Interrupts

Exceptions

- Exceptions are usually **synchronous and always tied to an assembly instruction**.
- An exception can arise at any stage of the execution of an instruction.

For example,

- ✓ **Illegal Instruction:** The hardware may detect **a bad opcode field during the instruction decode stage**. This will trigger an "illegal instruction" exception.
- ✓ **Instruction address misaligned:** An Instruction address misaligned exception is raised if the target address is **not aligned on a 4-byte or 2-byte boundary**, because the **core supports compressed instructions.**
- ✓ **Load/Store address misaligned**
- ✓ **Instruction/Load/Store access fault**
- ✓ Environment call

ecall	Environment Call	I	1110011	0x0	imm=0x0	Transfer control to OS
ebreak	Environment Break	I	1110011	0x0	imm=0x1	Transfer control to debugger

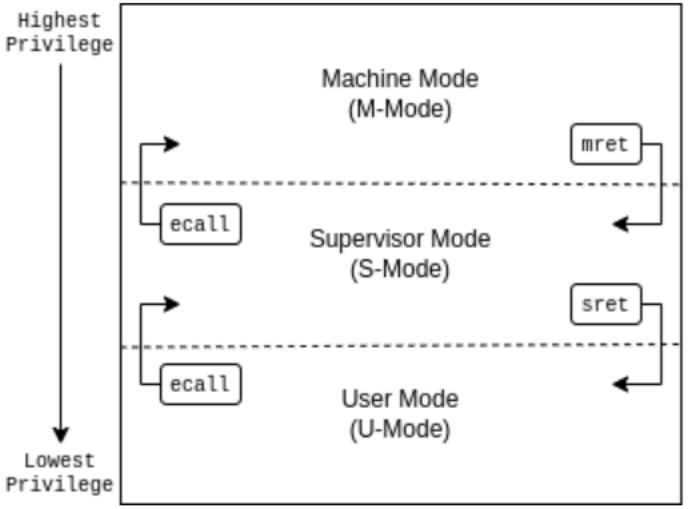
The Processor

Exceptions/Interrupts

Privilege Levels/ Modes

Privilege levels are used to provide protection between different components of the software stack, and **attempts to perform operations not permitted by the current privilege mode will cause an exception to be raised.**

Level	Encoding	Name	Abbreviation
0	00	User/Application	U
1	01	Supervisor	S
2	10	*Reserved*	
3	11	Machine	M



Implementations might provide anywhere from 1 to 3 privilege modes trading off reduced isolation for lower implementation cost,

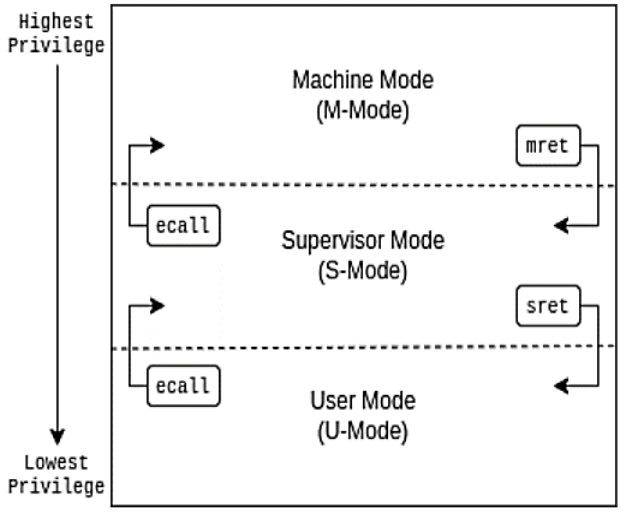
Number of levels	Supported Modes	Intended Usage
1	M	Simple embedded systems
2	M, U	Secure embedded systems
3	M, S, U	Systems running Unix-like operating systems

The Processor

Exceptions/Interrupts

Privilege Levels

1. **Machine mode (M-Mode)** - Allows unfettered access to all control and status level registers. **Any program running in M-Mode has virtually no restrictions.**
2. **Supervisor mode (S-Mode)** - **Allows the program to modify any supervisor-level control and status registers. S-Mode is designed for running a kernel**
3. **User mode (U-Mode)** - A process running **here cannot modify any of the hardware or the control and status registers. It must execute a system call so a higher privilege mode can do that action on its behalf.**



ecall	Environment Call	I	1110011	0x0	imm=0x0	Transfer control to OS
ebreak	Environment Break	I	1110011	0x0	imm=0x1	Transfer control to debugger

- Ecall is used to TRAP the processor
- Ecall can be used to switch between privilege Modes.
- Used to implement system call to pass/access system recourses i.e., Pass the arguments between different privileged modes

The Processor

Exceptions/Interrupts

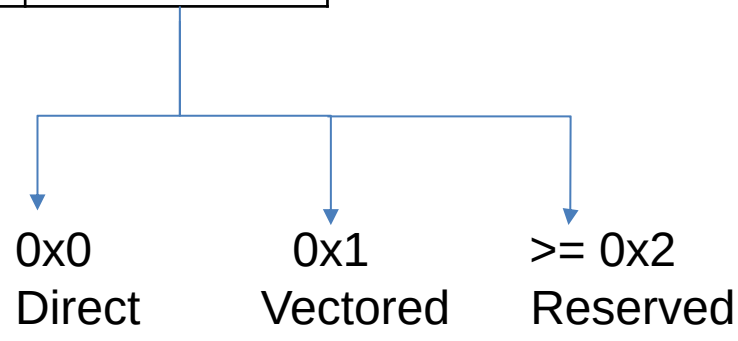
Direct and Vectored Mode of TRAP Handling

Two Modes used for TRAP Handling
Method 1: Direct Mode

mtvect : Machine TRAP Vector Register

mtvec[31:2]	mtvec[1:0]
Base Address [31:2]	Mode

Base Address [31:2] gives the base address of the Direct / Vectored Mode



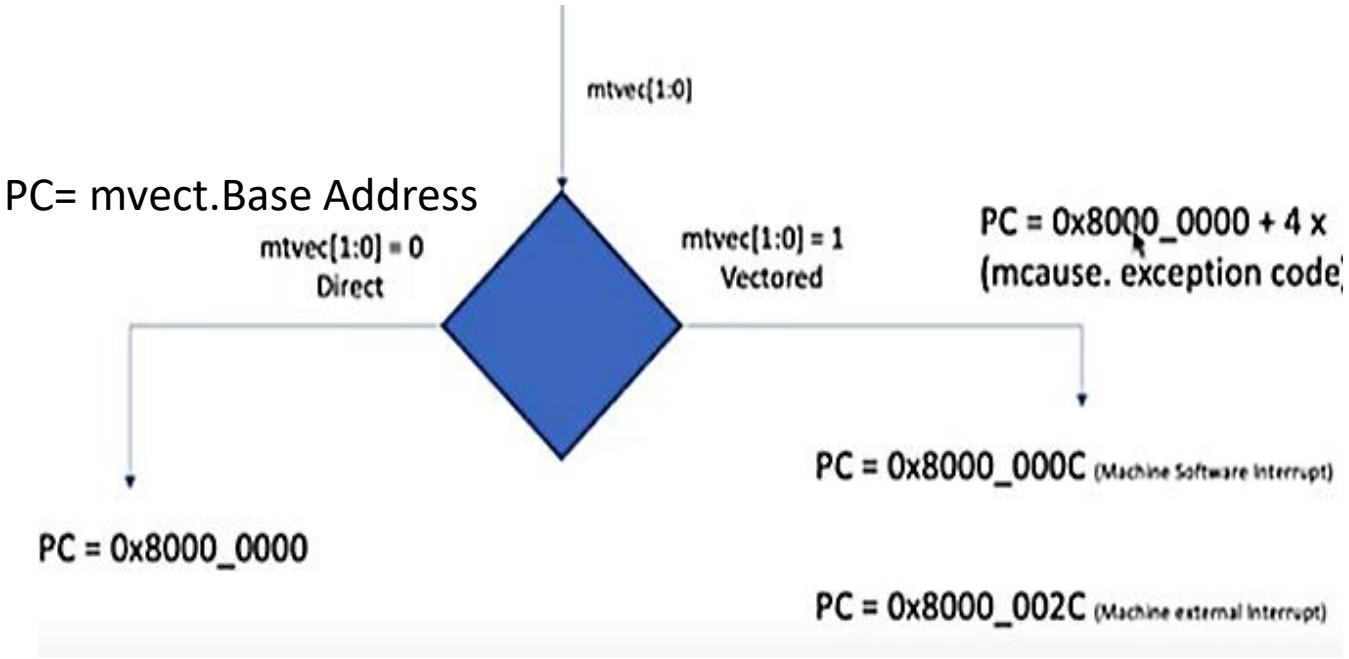
The Processor

Exceptions/Interrupts

Direct and Vectored Mode of TRAP Handling

Two Modes used for TRAP Handling
Method 1: Direct Mode

mtvect : Machine TRAP Vector Register



Direct Mode

- In this all-synchronous Exceptions and asynchronous Interrupts TRAP to mtvect.BASE address i.e.,
- PC= mtvect.BASE address
- Once control of execution changes to the TRAP Handler, Software must read **the scause / mcause register to determine what TRIGGERED the TRAP.**
- The *Supervisor Exception Cause Register* or *SCAUSE*)/ Machine cause Register (MCAUSE) are the registers used depending on privilege mode, to identify the reason for the exception.

The Processor

Exceptions/Interrupts

Direct and Vectored Mode of TRAP Handling

scause / mcause register

exception code helps to identify the type of exception

XLEN-1 Interrupt	XLEN-2 : 0 Exception Code (**WLRL**)	
1	XLEN-1	
0	0	Instruction address misaligned
0	1	Instruction access fault
0	2	Illegal instruction
0	3	Breakpoint
0	4	Load address misaligned
0	5	Load access fault
0	6	Store/AMO address misaligned
0	7	Store/AMO access fault
0	8	Environment call from U-mode
0	9	Environment call from S-mode
0	10	*Reserved*

1	0	User software interrupt
1	1	Supervisor software interrupt
1	2	*Reserved for future standard use*
1	3	Machine software interrupt
1	4	User timer interrupt
1	5	Supervisor timer interrupt
1	6	*Reserved for future standard use*
1	7	Machine timer interrupt
1	8	User external interrupt

- When an exception/interrupt happens, the hardware sets the **mcause register with the corresponding exception** code.
- The pc is **set to the trap handler base address**.
- The **exception code helps to identify the type of exception**.

The Processor

Exceptions/Interrupts

Direct and Vectored Mode of TRAP Handling

- Read cause, and transfer to relevant handler
- Determine action required
- **If restartable**
  - ✓ **Take corrective action**
  - ✓ use SEPC to return to the program
- **Otherwise**
  - ✓ **Terminate program**
  - ✓ Report errors using CSR registers

The Processor

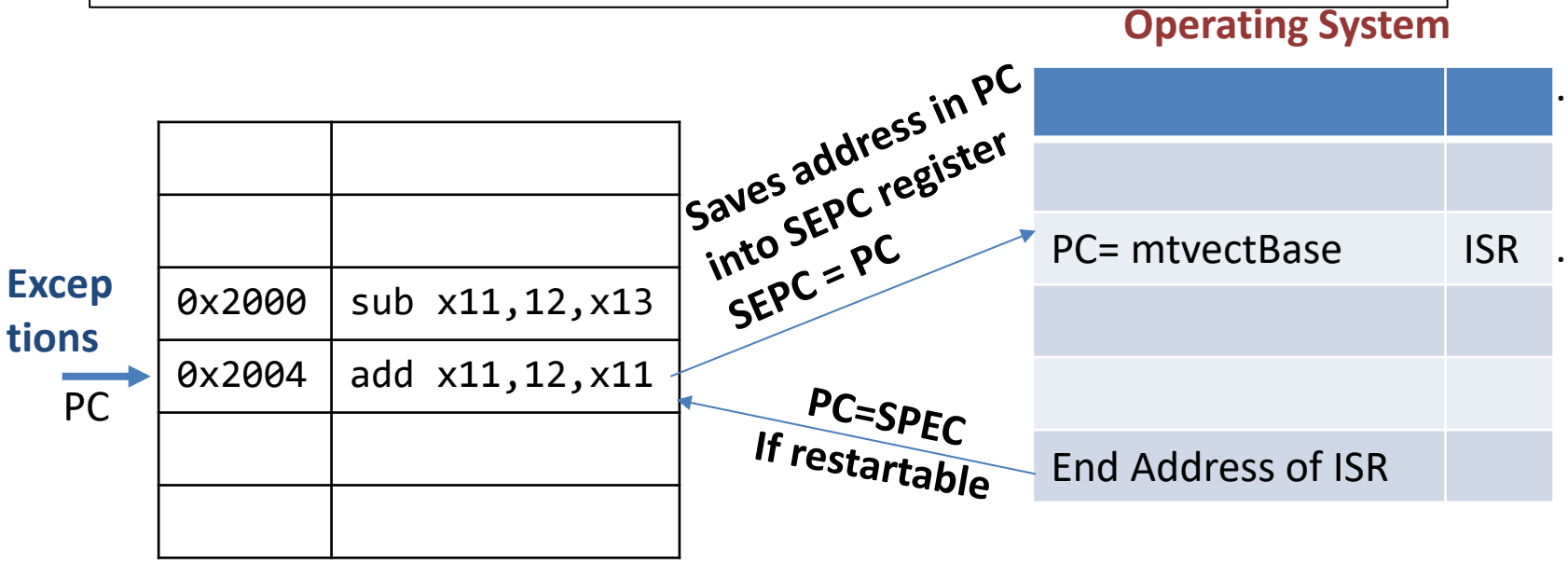
Exceptions

Direct and Vectored Mode of TRAP Handling

Exceptions that Current implementation can generate are

a) Execution of an undefined instruction or
b) A hardware malfunction.

Let us assume that a hardware malfunction occurs during the instruction
add x11, x12, x11



The basic action that the processor when an exception occurs is

1. To save the address of the **unfortunate instruction** in the ***supervisor exception cause register*** **(SEPC)** and
. then **transfer control to the operating system at some specified address.**
. The operating system can then take the appropriate action, (provides service by taking some predefined action) in **response to a malfunction, or stopping the execution of the program and reporting an error based on scause.**

The Processor

Exceptions

Direct and Vectored Mode of TRAP Handling

Method 2: Vectored Interrupts

In a vectored interrupt, the address to which control is transferred is determined by

- **PC (Vectored Exception Handler Address) = mtVect.Base + 4 x Exception/ Interrupt Code**
- For example, If it is illegal Instruction has caused TRAP then MCAUSE will be updated with MSB (Interrupt=0) and Exception Code =02. In this case
  PC = mtVect.Base + 4 x 02 = 0x8000 0000 + 0x10 (=16d) = 0x8000 0010
- If it machine Time Interrupt the PC=0x8000 0000 + 4 x 7 =0x8000 001C

PC (Vectored Exception Handler Address) = mtVect.Base + 4 x Exception/ Interrupt Code

Vector Table	
mtvect_Base address	0x8000 0000
	.....

The Processor

Exception/Interrupt CSR Registers

Interrupt Enable / Disable: Each of the Timer, Software, and External Interrupts **can be enabled globally/ individually.**

- **Globally, all the interrupts can be enabled/disabled using the MIE bit in the SSTATUS/MSTATUS register. Ex: MIE=1 Machine Mode Interrupt is emabled**

- **Machine Interrupt Enable (mie) Register: The MTIE, MSIE, and MEIE bit enable's/disable's Timer, Software, and External interrupts individually.**
- **Ex: If MTIE=0; Machine TIMER Interrupt is disabled**

- **Machine Interrupt Pending (mip) Register: The bits in mip indicates which Interrupts TRAPs pending to be served.** MEIP is read-only in mip, and is set and cleared by a platform-specific interrupt controller.

15 ... 12	11	10	9	8	7	6	5	4	3	2	1	0
0	MEIP	0	SEIP	0	MTIP	0	STIP	0	MSIP	0	SSIP	0
4	1	1	1	1	1	1	1	1	1	1	1	1

Figure 3.14: Standard portion (bits 15:0) of mip.

15 ... 12	11	10	9	8	7	6	5	4	3	2	1	0
0	MEIE	0	SEIE	0	MTIE	0	STIE	0	MSIE	0	SSIE	0
4	1	1	1	1	1	1	1	1	1	1	1	1

Figure 3.15: Standard portion (bits 15:0) of mie.

the interrupt-pending

interrupt-enable bits

The Processor

Exceptions in a Pipelined Implementation

How Pipelined Implementations handle Exceptions ?

- Pipelined Implementations treat Exceptions as **another form of control hazard**
- For example, Suppose there is a hardware malfunction while executing add in EX stage i.e., **add x1, x2, x1 in EX stage means next 2 instructions are there is ID and IF stage respectively**

  - ✓ Prevent x1 from being clobbered (Not updated)
  - ✓ Complete previous instructions
  - ✓ **Flush add and subsequent instructions which are in ID and IF**
  - ✓ Set SEPC and SCAUSE register values
  - ✓ Transfer control to handler

- Similar to mispredicted branch
  - Use much of the same hardware

We will use the same mechanism we used for taken branches, but this time the exception causes the deasserting of control lines.

Exception in a Pipelined Computer

Given this instruction sequence,

```
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   lw     x16, 100(x7)
. . .
```

assume the instructions to be invoked on an exception begin like this:

```
1C090000hex   sw    x26, 1000(x10)
1C090004hex   sw    x27, 1008(x10)
. . .
```

The Processor

Exceptions in a Pipelined Implementation

Exception in a Pipelined Computer

Given this instruction sequence,

```
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   lw     x16, 100(x7)
. . .
```

assume the instructions to be invoked on an exception begin like this:

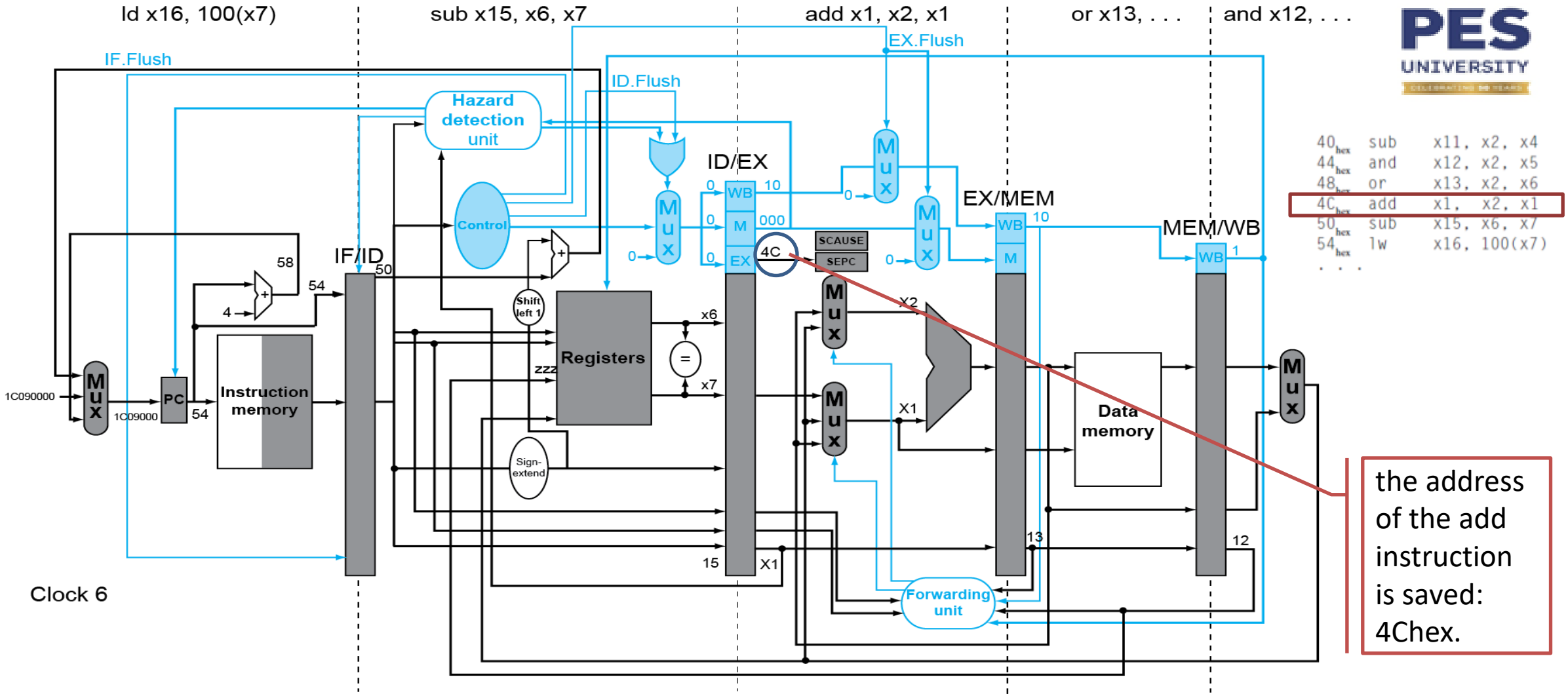
```
1C090000hex   sw    x26, 1000(x10)
1C090004hex   sw    x27, 1008(x10)
. . .
```

Show what happens in the **pipeline if a hardware malfunction exception occurs when add instruction is in EX stage of the pipeline**

The Processor

Exceptions in a Pipelined Implementation

ADD Instruction in EX stage. Assume the hardware **malfunction is detected during that phase**, and **0000 0000 1C09 0000hex** is forced into the PC

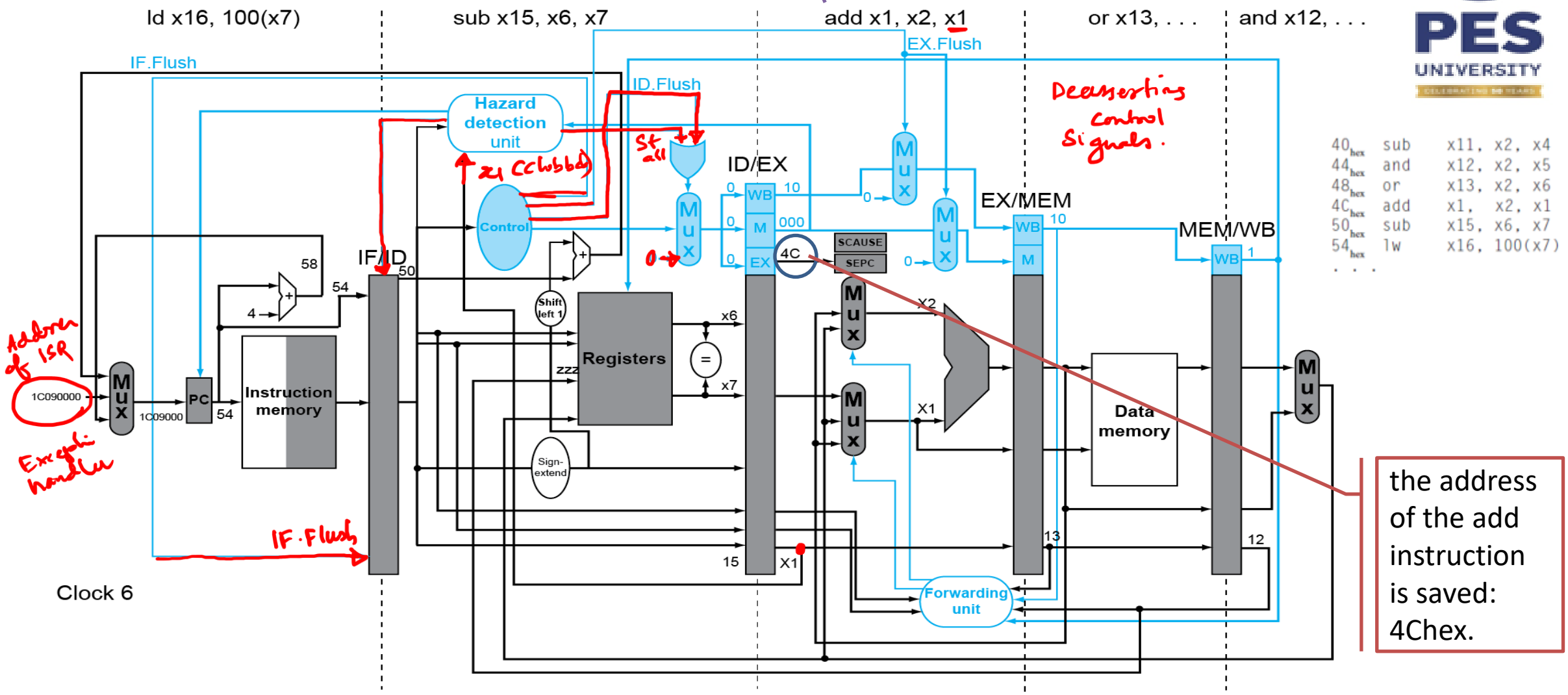


the address of the add instruction is saved: 4Chex.

The Processor

Exceptions in a Pipelined Implementation

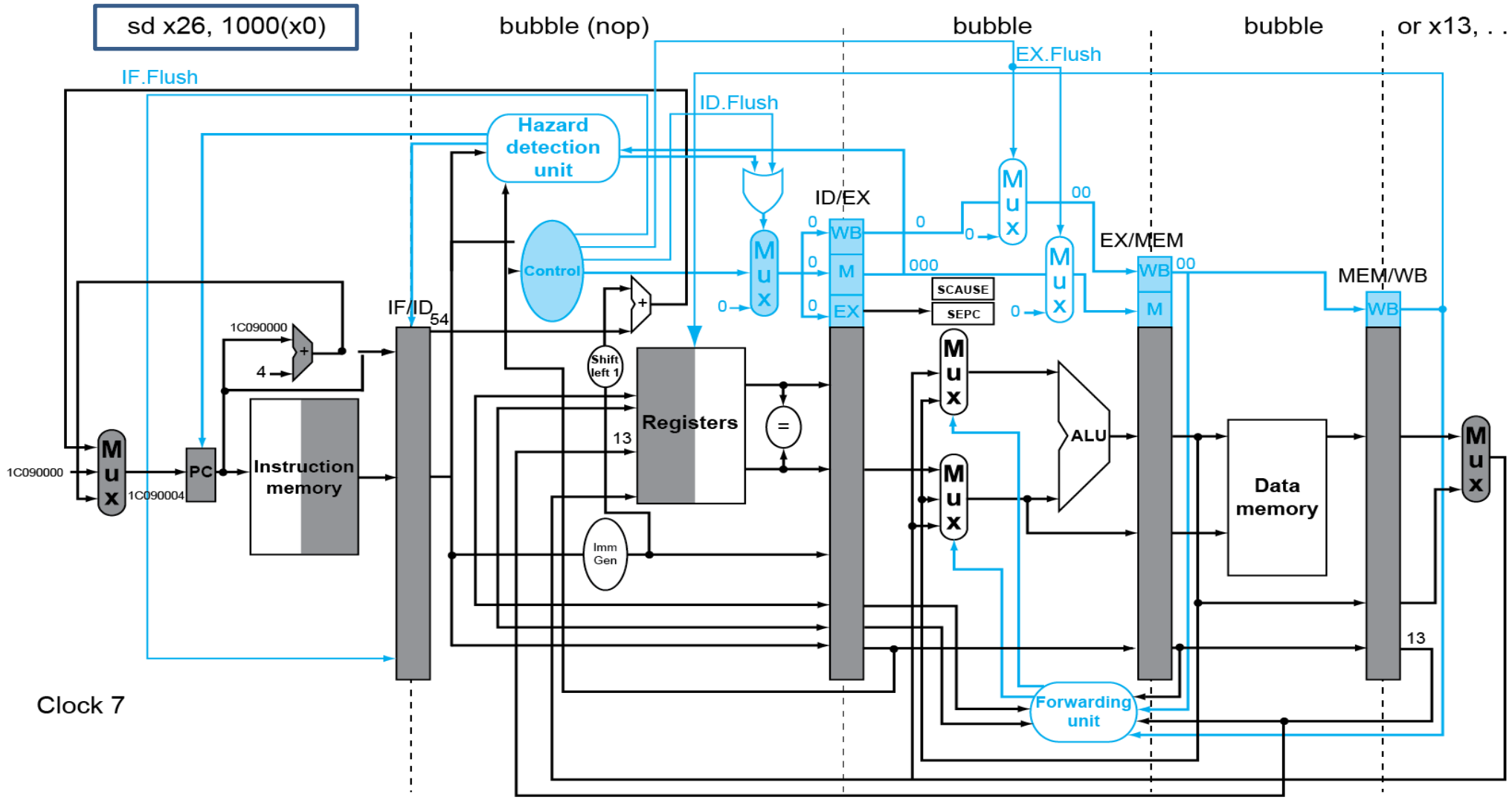
ADD Instruction in EX stage. Assume the hardware **malfunction is detected during that phase**, and **0000 0000 1C09 0000hex** is forced into the PC



the address of the add instruction is saved: 4Chex.

The Processor

Exceptions in a Pipelined Implementation



Clock cycle 7 shows that the **add and following instructions are flushed**, and the **first instruction of the exception** handling code is fetched.

The Processor

Exceptions in a Pipelined Implementation

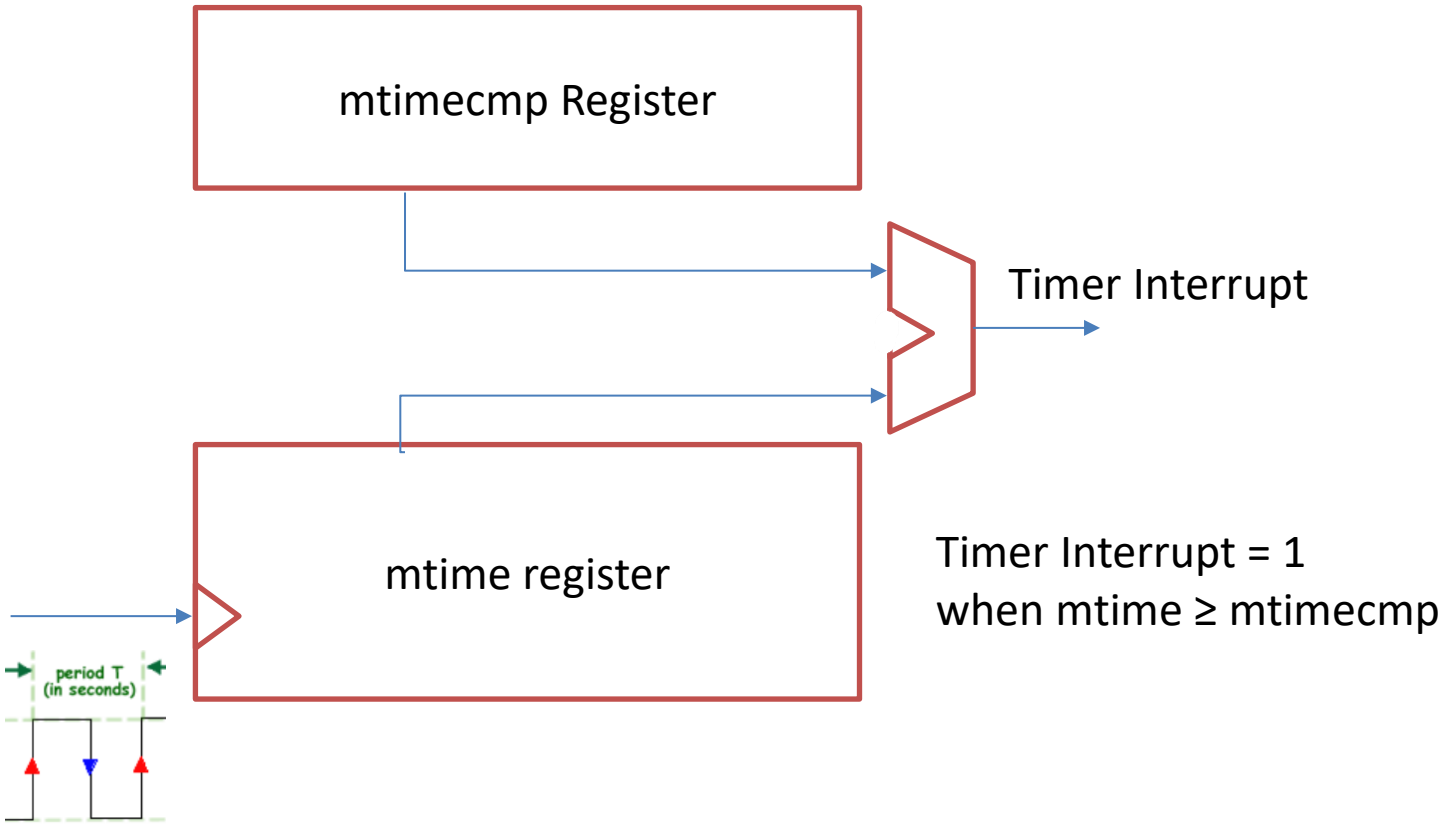
Multiple Exceptions

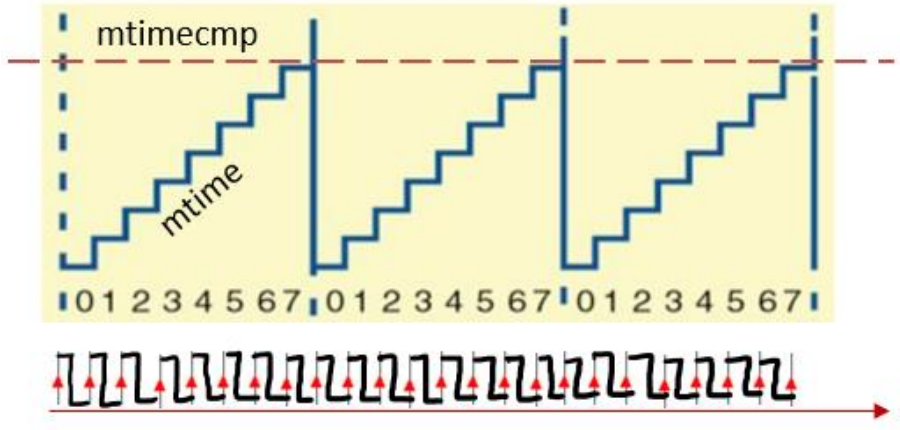
- Pipelining overlaps multiple instructions
  - ✓ Could have multiple exceptions at once
- Simple approach: deal with exception from earliest instruction
  - ✓ Flush subsequent instructions
  - ✓ “Precise” exceptions
- In complex pipelines
  - ✓ Multiple instructions issued per cycle
  - ✓ Out-of-order completion
  - ✓ Maintaining precise exceptions is difficult!

The Processor

Timer Interrupt

- A “timer interrupt” is caused when a separate timer circuit indicates that a predetermined interval has ended.
- The timer subsystem will interrupt the currently executing code.





Timer Interrupt = 1
when mtime ≥ mtimecmp

The timer interrupts are handled by the OS which uses them to implement time-sliced multi-threading.

The Processor

Timer Interrupt

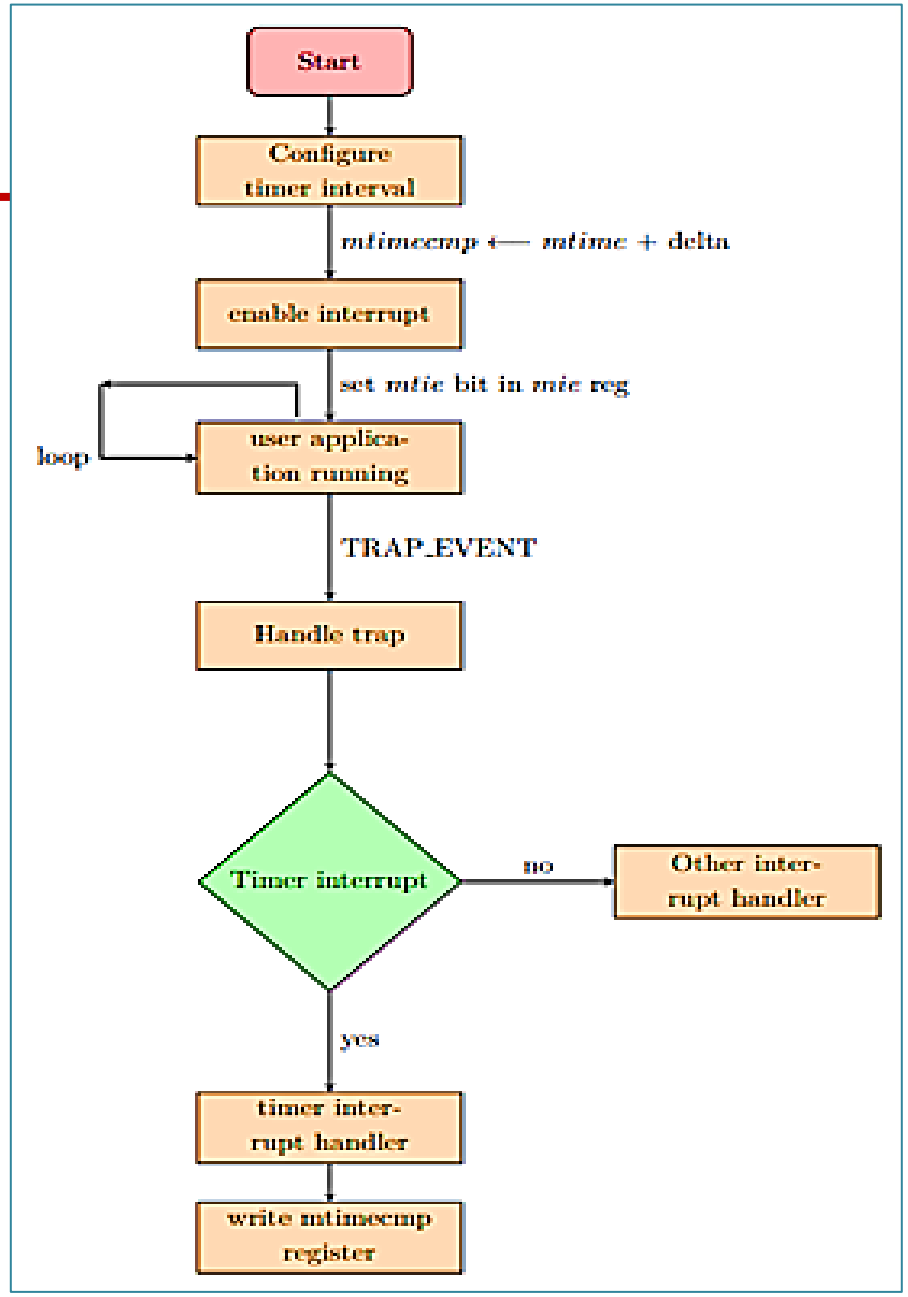
Timer Registers

mtime Register

- mtime register is a synchronous counter.
- It starts running from the time the processor is powered on and provides the current real-time in ticks.

mtimecmp Register

- This register is used to store the time period after which a timer interrupt should happen.
- The value of mtimecmp is compared with the mtime register.
- When the mtime value becomes greater than mtimecmp, a timer interrupt happens.
- Both the mtime and mtimecmp registers are 64-bit memory-mapped registers.



The Processor

Exceptions and Interrupts

5. Assume there is an illegal instruction getting executed, what value will be updated in the **scause** register?
   Note: Use the table given

Interrupt bit	Exception Code	
0	0	Instruction address misaligned
0	1	Instruction access fault
0	2	Illegal instruction
0	3	Breakpoint
0	4	Load address misaligned
0	5	Load access fault
0	6	Store/AMO address misaligned
1	0	User software interrupt
1	1	Supervisor software interrupt
1	2	*Reserved for future standard use*
1	3	Machine software interrupt
1	4	User timer interrupt
1	5	Supervisor timer interrupt
1	6	*Reserved for future standard use*
1	7	Machine timer interrupt

   a. Interrupt bit =1 and Exception Code=2
   b. Interrupt bit =0 and Exception Code=2
   c. Interrupt bit =0 and Exception Code=1
   d. Interrupt bit =1 and Exception Code=4
6. If the **scause** MSB is 1 what is the cause for the trap?
   a. Interrupt
   b. Exception
   c. Both
   d. None of these
7. Which of the following is not an Exception?
   a. System Reset
   b. Invoking the OS from use program
   c. Using undefined Instruction
   d. I/O device request

The Processor

Exceptions and Interrupts

8. Which of the following is either an Exception/ Interrupt?
   a. I/O device request
   b. Using undefined Instruction
   c. I/O device request
   d. Hardware Malfunctions
9. Interrupts are unexpected events that occur _____any of the RISC-V harts.
   a. Asynchronously Inside
   b. Synchronously outside
   c. Asynchronously outside
   d. None of these
10. Which of the following modes has the highest privilege?
   a. Supervisor mode (S-Mode)
   b. Machine mode (M-Mode)
   c. User mode (U-Mode)
   d. All of them
11. Which of the following modes is a must to have in a RISC-V SoC?
   a. Supervisor mode (S-Mode)
   b. Machine mode (M-Mode)
   c. User mode (U-Mode)
   d. All of them

The Processor

Exceptions and Interrupts

12. ecall is an instruction used to _____
    a. switch between privilege Modes
    b. transfer control to debugger
    c. transfer control to subroutine
    d. None of these
13. In the case of the Vectored Mode of TRAP Handling, control of execution transfer to the handler by updating PC with _____
    a. VT_BaseAddress + 4 x Exception Code from scause
    b. VT_BaseAddress
    c. VT_BaseAddress + Exception Code from scause
    d. None of these
14. In which of the following TRAP handlers the SEPC is copied into PC to restart the TRAPPED program
    a. User timer Interrupt TRAP handler
    b. Illegal Instruction TRAP handler
    c. Instruction addresses misaligned TRAP handler
    d. None of these
15. Which Timer register provides the current real-time in ticks?
    a. mtimecmp
    b. mtime
    c. Both mtime and mtimecmp
    d. None of these

The Processor

Exceptions and Interrupts

16. Which Timer register is used to hold the Initial count as reference w.r.t time period to generated?
    a. mtimecmp
    b. mtime
    c. Both mtime and mtimecmp
    d. None of these
17. Under which of the following conditions the timer Interrupt is invoked?
    a. mtime<mtimecmp
    b. mtime>=mtime+delta
    c. mtime=stime
    d. None of these
18. mie register is used for _____
    a. holding the pending exception to handled
    b. enable/ disable the unexpected events using corresponding bits
    c. to globally enable/ disable the unexpected events using a single bit.
    d. None of these.
19. Which of the following privileged modes **has virtually no restrictions**?
    a. Machine
    b. Supervisor
    c. User
    d. None of these
20. Assume a RISC-V core running in User privilege mode. Which is a TRUE statement
    a. A process running in User mode cannot modify any of the control and status registers
    b. A process running in User mode can modify any of the control and status registers
    c. A process running in User mode can modify only supervisor-level control and status registers
    d. A process running in User mode can modify only machine-level control and status registers

The Processor

Exceptions and Interrupts

2. Consider the program given executed on a 5-stage pipelined architecture

```
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   lw     x16, 100(x7)
. . .
```

assume the instructions to be invoked on an exception begin like this

```
1C090000hex   sw    x26, 1000(x10)
1C090004hex   sw    x27, 1008(x10)
. . .
```

Answer the following assuming a hardware malfunction exception has been invoked when the add instruction is in EX stage of the pipeline

a) What will be the content of SEPC
b) What will be the status of Interrupt bit in SCAUSE?
c) Where is the control of execution transferred?
d) Which instructions in the pipeline get flushed & gets executed?

3. Explain the steps involved in handling the TRAP using the Vector Table approach.
4. Explain with a flow chart how a timer can be used in Interrupt mode to generate the time delay.

RISC V Architecture

Mahesh Awati
Department of Electronics and Communication Engg.
Electronic City Campus