Interrupt Cont.

EECS 388 – Spring 2023

Lecture notes based in part on slides created by Dr. Mohammad Alian and Dr. Heechul Yun

Review

- What are the disadvantages of polling for managing I/O devices? Select all that apply.

1. Frequent polling prevents the CPU from running non-I/O operations
2. Cannot be used for multiple I/O devices
3. Infrequent polling may miss inputs from I/O

Review

- An interrupt occurs during the decode stage of an instruction in the running program. When the program execution will jump to the interrupt service routine?

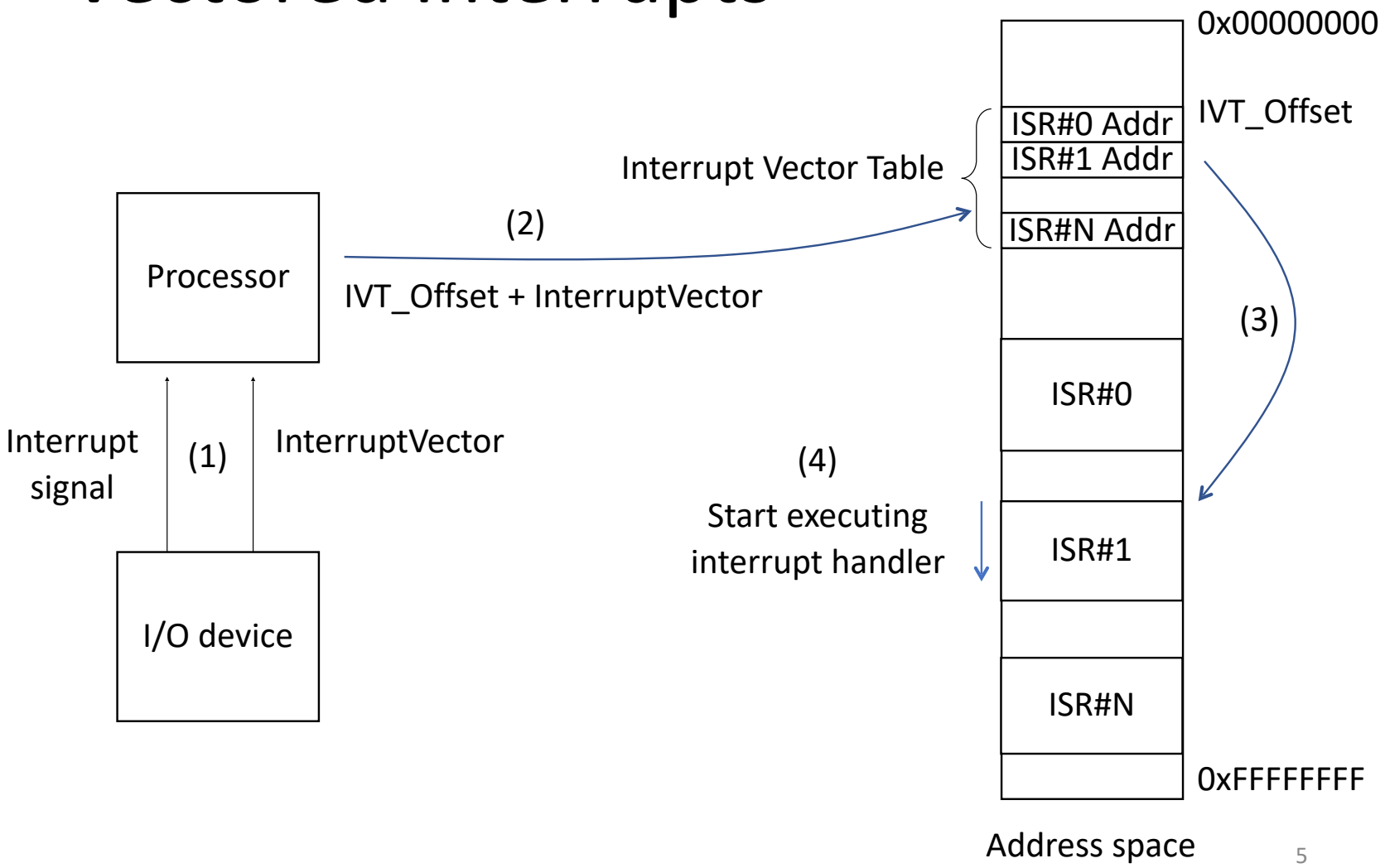
1. Right after completing the Decode stage of the current instruction cycle
2. After completing all stages necessary to fully execute the current instruction
3. Right after completing the Execute stage of the current instruction cycle

Vectored Interrupts

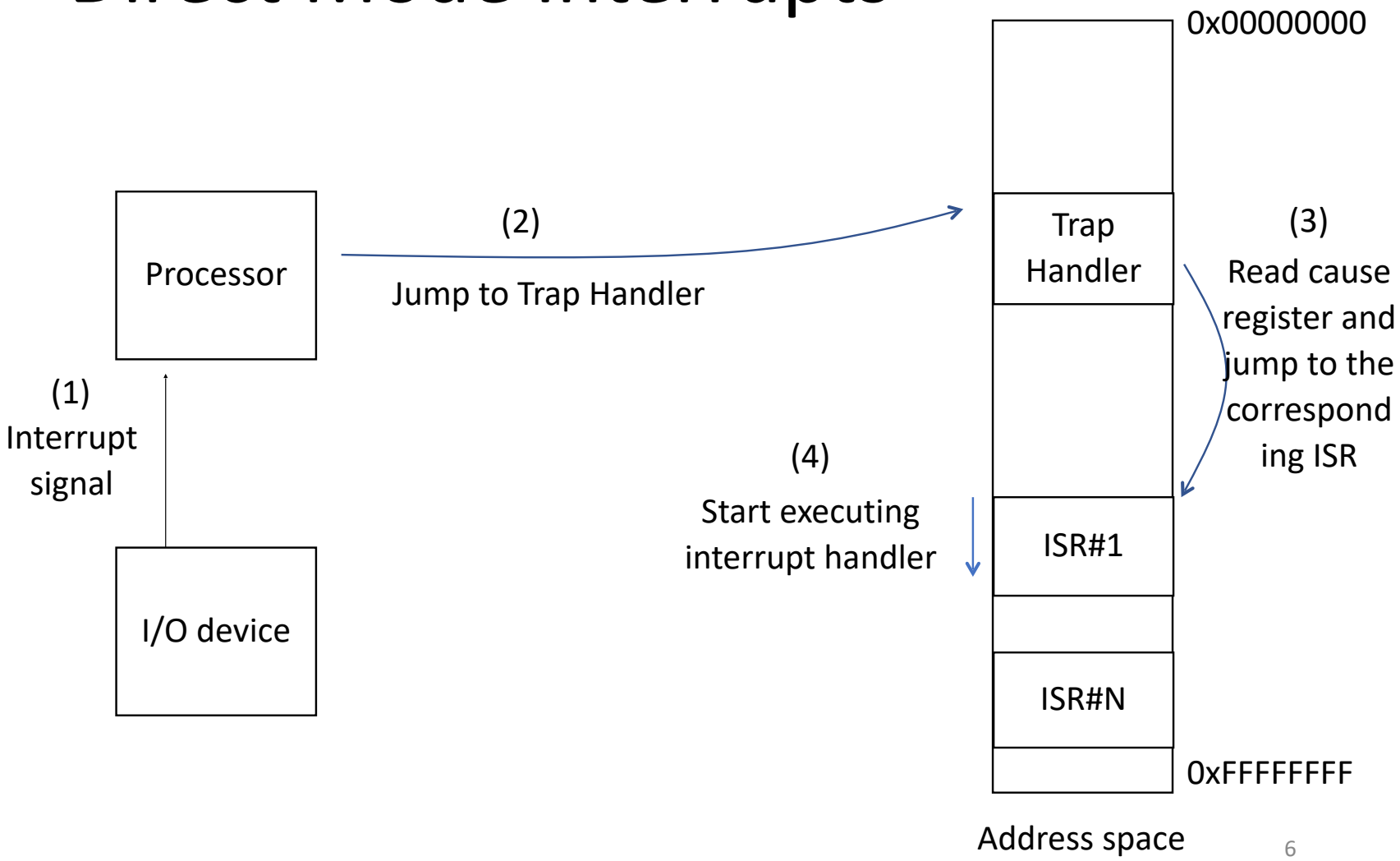
- An ***interrupt vector*** is provided to the processor by the interrupting device
- The *interrupt vector* is use by the processor to reference an entry in the ***Interrupt Vector Table***
- *Interrupt Vector Table* consists of memory locations each containing the starting address of an ***Interrupt Service Routine***
- *Interrupt Service Routines* are program fragments stored in memory that service interrupt requests

4

Vectored Interrupts



Direct Mode Interrupts



Q1

Suppose we are asked to write a program that takes a sequence of 100 characters typed on a keyboard and processes the information contained in those 100 characters.

- Assume the characters are typed at the rate of one character every 0.125 seconds.
- Assume the processing of the 100-character sequence takes 12.4999 seconds.

How much "CPU time" will it take to complete the task with a polling-driven keyboard?

Total time=time to receive the data + time to process the data

7

- **Solution Q1:**
- For polling driven I/O:
- CPU does not know when the data is coming, so it polls constantly
- Time needed to receive 100-character sequence

  =100 x 0.125sec=12.5 seconds

Total time = Time to read+ time to process
=12.5+12.4999

=24.9999 seconds

8

In class Quiz 7

Assume an interrupt-driven keyboard that takes 100ns for executing the ISR (handling an interrupt). How much "CPU time" does it take for the interrupt-driven Keyboard to a sequence of 100 characters? How much time we save by doing interrupt?

Solution Q2:

Total time for reading 100 character=

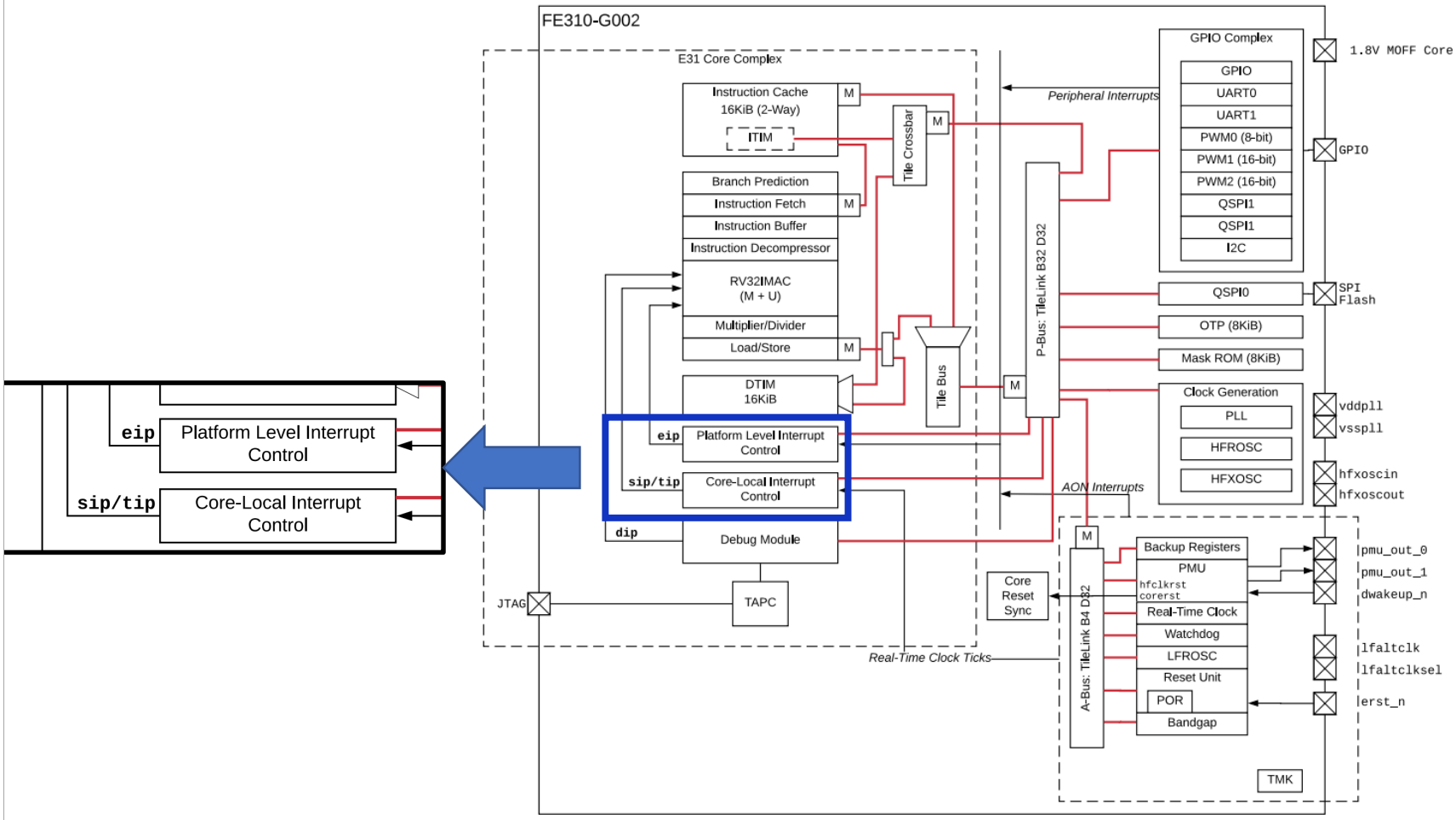
Total time= Time to read+ time to process

Therefore, compared to polling we are saving=

Which part of the polling making it slow?

9

Case Study: Si-Five Interrupts



Source: SiFive FE310-G002 Manual

10

Si-Five Interrupt Architecture

Based on the source of interrupts, there are two main forms of interrupt in the Si-Five processor.

- Local and global (external) interrupts

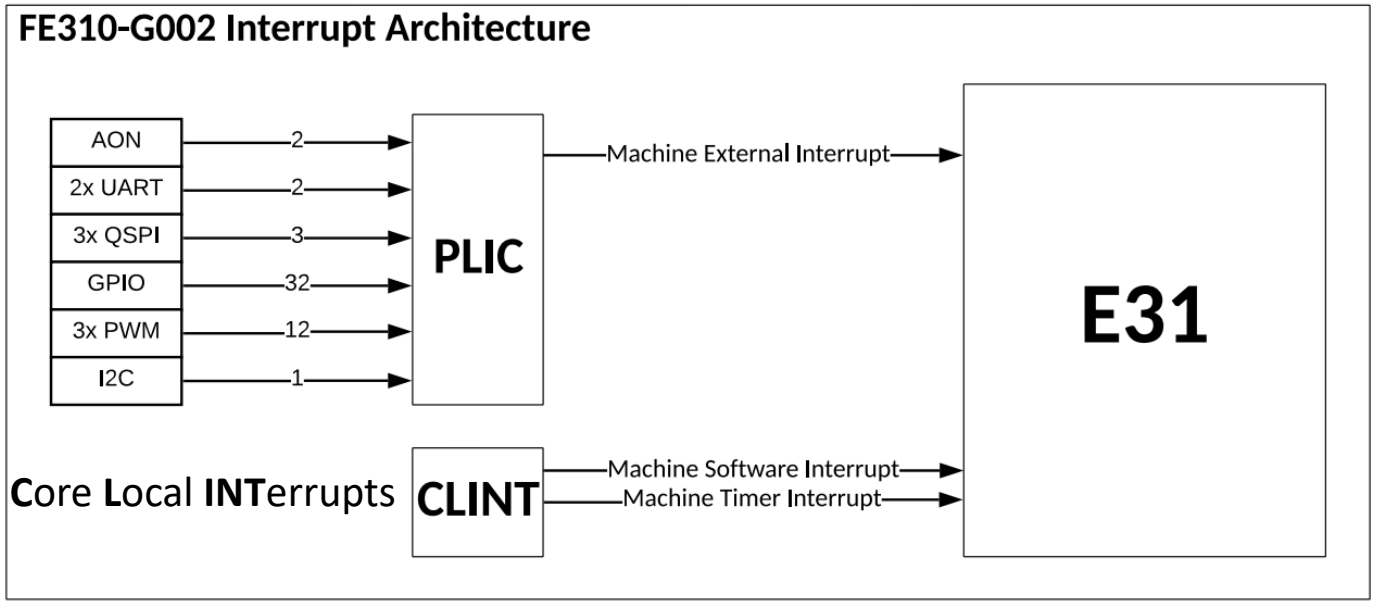


Figure 4: FE310-G002 Interrupt Architecture Block Diagram.

Source: Si-five Programmer's Manual

11

Local and global interrupts

- The local interrupts are generated based on some activity internal to the processor.
  - They can be generated using software running in the processor or a timer that is counting towards a certain number.
  - Software interrupts are most useful for inter-processor communication
- To handle the local interrupts the Si-Five processor has a module called CLINT or **C**ore **L**ocal **INT**errupts

12

Local and global interrupts

- The global or external interrupts are interrupts from external devices such as GPIO, UART, I2C etc.
- To handle the global interrupts the Si-Five processor has a module called PLIC or platform-level interrupt controller
  - PLIC is connected to GPIO, I2C, Etc.
  - PLIC supports 52 global interrupts with 7 priority level
  - We studied in last class that to combine the interrupt from multiple IO devices, we need priority encoding.
  - The PLIC module helps in this process.

13

Control and Status Registers (CSRs)

- Privilege registers for software/hardware communication
- Use special instructions to read/write

hart = HARdware **T**hread

Interrupt related CSRs:

`mstatus`: used to enable or disable global interrupt

`mie`: enable/disable individual interrupts

`mip`: which interrupts are currently pending

`mtvec`: base address of the interrupt vector

`mepc`: used for storing the PC before handling the interrupt

`mcause`: shows the cause of the interrupt

14

Machine Status Register (`mstatus`)

- The `mstatus` register keeps track of and controls the hart's current operating state,
  - including whether or not interrupts are enabled.
  - Interrupts are enabled by setting the MIE bit in mstatus and by enabling the desired individual interrupt in the `mie` register (a different register shown in next slide)

Machine Status Register			
CSR	`mstatus`		
Bits	**Field Name**	**Attr.**	**Description**
[2:0]	Reserved	WPRI	
3	MIE	RW	Machine Interrupt Enable
[6:4]	Reserved	WPRI	
7	MPIE	RW	Machine Previous Interrupt Enable
[10:8]	Reserved	WPRI	
[12:11]	MPP	RW	Machine Previous Privilege Mode

hart = HARdware Thread

Machine Interrupt Enable (`mie`)

- We have two types of local and one global interrupt
- Individual interrupts are enabled by setting the appropriate bit in the `mie` register

Machine Interrupt Enable Register			
CSR	`mie`		
Bits	**Field Name**	**Attr.**	**Description**
[2:0]	Reserved	WPRI	
3	MSIE	RW	Machine Software Interrupt Enable
[6:4]	Reserved	WPRI	
7	MTIE	RW	Machine Timer Interrupt Enable
[10:8]	Reserved	WPRI	
11	MEIE	RW	Machine External Interrupt Enable
[31:12]	Reserved	WPRI	

16

Machine Interrupt Pending (`mip`)

- The system might be dealing with one interrupt while others are pending
- This register provides a way to know what is pending

Machine Interrupt Pending Register			
CSR	`mip`		
Bits	**Field Name**	**Attr.**	**Description**
[2:0]	Reserved	WIRI	
3	MSIP	RO	Machine Software Interrupt Pending
[6:4]	Reserved	WIRI	
7	MTIP	RO	Machine Timer Interrupt Pending
[10:8]	Reserved	WIRI	
11	MEIP	RO	Machine External Interrupt Pending
[31:12]	Reserved	WIRI	

17

Machine Trap Vector (`mtvec`)

- The mtvec register has two main functions:
  - defining the base address of the trap vector
    - See interrupt vector in slide 3
  - and setting the mode (direct or vector) by which the sifive will process interrupts.
    - See direct and vectored interrupt in slide 3

Machine Trap Vector Register			
CSR	`mtvec`		
Bits	**Field Name**	**Attr.**	**Description**
[1:0]	MODE	WARL	`MODE` Sets the interrupt processing mode. The encoding for the FE310-G002 supported modes is described in Table 19.
[31:2]	BASE[31:2]	WARL	Interrupt Vector Base Address. Requires 64-byte alignment.

MODE Field Encoding `mtvec.MODE`		
Value	**Name**	**Description**
0x0	Direct	All exceptions set `pc` to `BASE`
0x1	Vectored	Asynchronous interrupts set `pc` to `BASE` + 4 × `mcause.EXCCODE`.
≥ 2	Reserved	

18

Machine Cause (`mcause`)

- There many reason a system is interrupted (local/global interrupts and exceptions)
  - Broadly they are known as TRAP
- `mcause` Indicates the event that caused the trap
- If Trap is due to interrupt, MSB=1
- `mcause is` Similar to cause register in direct mode (slide 4)

Machine Cause Register			
CSR	`mcause`		
Bits	**Field Name**	**Attr.**	**Description**
[9:0]	Exception Code	WLRL	A code identifying the last exception.
[30:10]	Reserved	WLRL	
31	Interrupt	WARL	1 if the trap was caused by an interrupt; 0 otherwise.

19

Machine Cause (`mcause`)

Machine Cause Register			
CSR	`mcause`		
Bits	**Field Name**	**Attr.**	**Description**
[9:0]	Exception Code	WLRL	A code identifying the last exception.
[30:10]	Reserved	WLRL	
31	Interrupt	WARL	1 if the trap was caused by an interrupt; 0 otherwise.

Interrupt Exception Codes		
Interrupt	**Exception Code**	**Description**
1	0–2	Reserved
1	3	Machine software interrupt
1	4–6	Reserved
1	7	Machine timer interrupt
1	8–10	Reserved
1	11	Machine external interrupt
1	≥ 12	Reserved
0	0	Instruction address misaligned
0	1	Instruction access fault
0	2	Illegal instruction
0	3	Breakpoint
0	4	Load address misaligned

Interrupt Priorities

- We discussed priorities in last class
- In Sifive there are there are the local and global interrupts have different priorities
- Sifive interrupts are prioritized as follows, in decreasing order of priority:
  - Machine external/global interrupts
  - Machine software interrupts
  - Machine timer interrupts
- Individual priorities of global interrupts are determined by the PLIC

21

Setting external/global interrupts

- Each PLIC interrupt source can be assigned a priority by writing to its 32-bit memory-mapped `priority` register
- supports 7 levels of priority (0 being lowest)

PLIC Interrupt Priority Register (`priority`)				
Base Address		0x0C00_0000 + 4 × Interrupt ID		
Bits	**Field Name**	**Attr.**	**Rst.**	**Description**
[2:0]	Priority	RW	X	Sets the priority for a given global interrupt.
[31:3]	Reserved	RO	0	

22

References

- Patel, S., and Yale Patt. *Introduction to Computing Systems: from bits & gates to C & beyond*. McGraw-Hill Professional, 2019.

- https://static.dev.sifive.com/E31-RISCVCoreIP.pdf

23