Exceptions & Interrupts

Trap?

- Trap is a specific scenario caused by a exceptional condition or interrupt.
- In RISC-V, the term trap refers to, transfer of control to a trap handler caused either by an exception or an interrupt.
- Exception is an unusual condition occurring at run time of an instruction in the current RISCV hart.

Exceptions:

- Exceptions are usually synchronous and always tied to an assembly instruction. A exception can arise at any stage of execution of an instruction.
- For example, during instruction decode stage, the hardware may detect a bad opcode field. This will trigger a “illegal instruction” exception.
- The exception code helps to identify the type of exception. The possible exceptions in RISC-V are listed below:
- • Illegal instruction
- • Instruction/Load/Store address misaligned
- • Instruction/Load/Store access fault
- • Environment call
- • Break point

1. Illegal Instruction Exception:

The exception occurs when the programs tries to execute any illegal instruction. For example trying to write on a read-only CSR register will generate a illegal instruction exception.

Example:

```
li t0, 8            # t0 ←– 8
csrrs x0, mhartid,      t0 # Attempt to write to a read-only CSR, generates exception
```

2. Instruction Address Misaligned Exception

The exception occurs when the programs tries to execute an unconditional jump or take a branch,

wherein the target address is not 4 byte aligned. For example, executing a program with start address as 0x80000001. This will generate a instruction address misalignment exception on a unconditional jump.

Example:

```
# start address set to 0x80000001 ( start not aligned to 4 byte boundary.
start: la x15, loop              # x15 ←− Address (loop)
jalr ra, x15 ,0              # Jumping to a label (loop) which is not 4 byte aligned
                                 # This causes an Instruction address misalignment exception
loop: addi x10, x10,1    # x10 ←− x10+1
```

3 .Load Address Misaligned Exception

- The exception occur when the programs tries to execute an load instruction to access data from misaligned address or an address that is not 4 byte aligned. For example, trying to access a data section without using a properly aligning it would cause this exception.
- example:

4.Store Address Misaligned Exception:

The exception occurs when the programs tries to execute an store instruction at a misaligned address (Address that is not four byte aligned). For example trying to store data into a data section without using proper alignment, would cause this exception.

example:

5.Instruction Access Fault:

The exception occurs when the programs tries to access an instruction on a invalid memory location.

For example executing unconditional jump instruction to a memory location which is out of bounds of the physical memory

example:

6.Load Access Fault

The exception occurs when the programs attempt to do a load on a invalid memory location. For example trying to load from address which is more than the bound of memory or inaccessible by memory.

7.Store Access Fault:

The exception occurs when the programs attempts to do a store on an invalid memory location.

For example, trying store to address which is more than the bound of memory or inaccessible by memory

Example:

8.Break Point

The exception occurs when the programs executes a break-point set in the program to enter debug mode.

9.Environment Call

This exception occurs when the programs executes a system call. The system call is realized in RISC-V using ecall instruction. The ecall instructions can also used to switch from lower privilege modes to higher privilege modes

example:

```
addi x10, x10, 2
ecall                # Environment call exception generated
```

Handling Exceptions

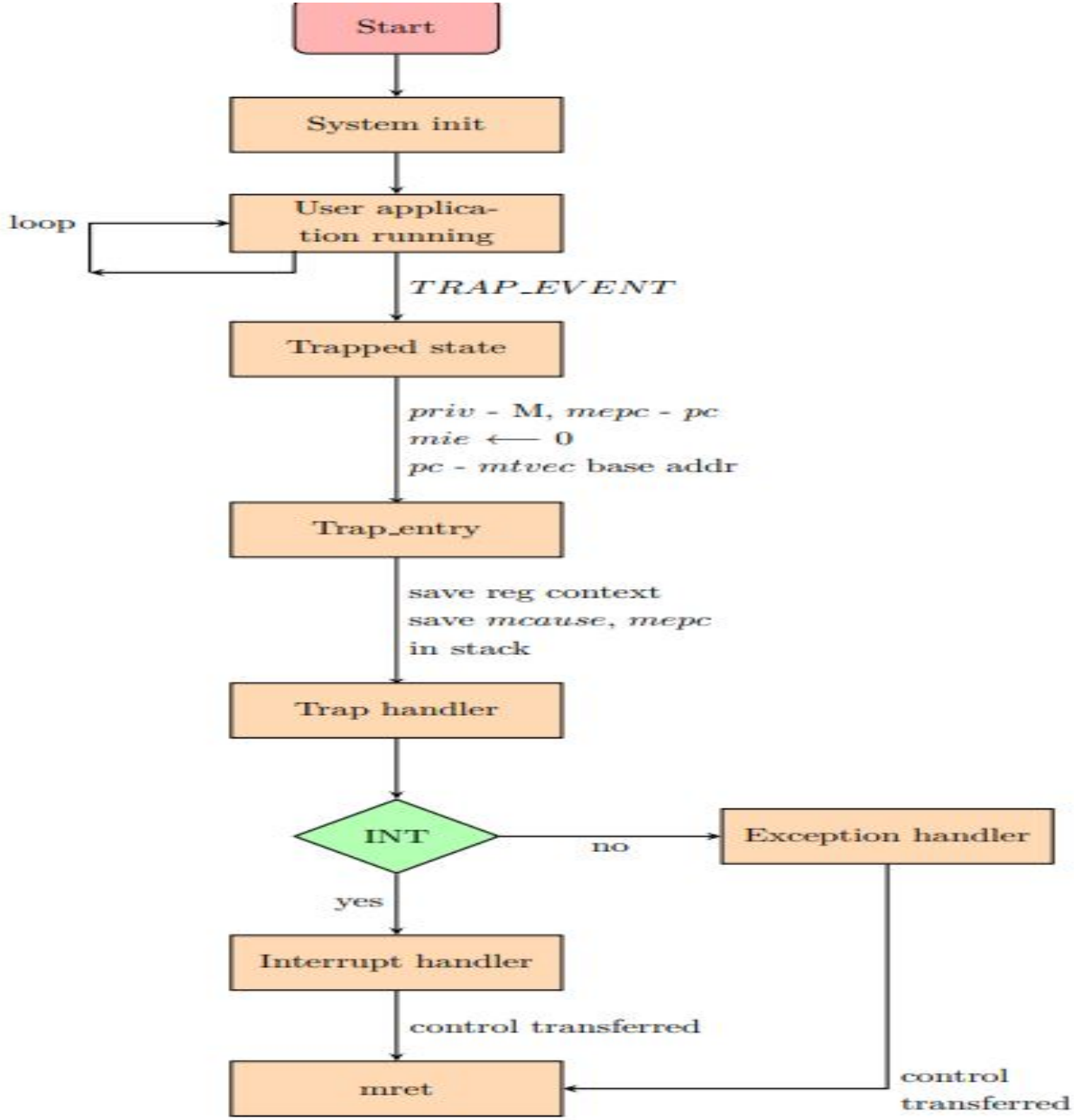
- Once an exception happens the processor stops execution and passes the control the trap handler.
- Inbetween this, the processor privilege is set to Machine mode and processor sets the mcause register with exception code
- When a trap occurs:

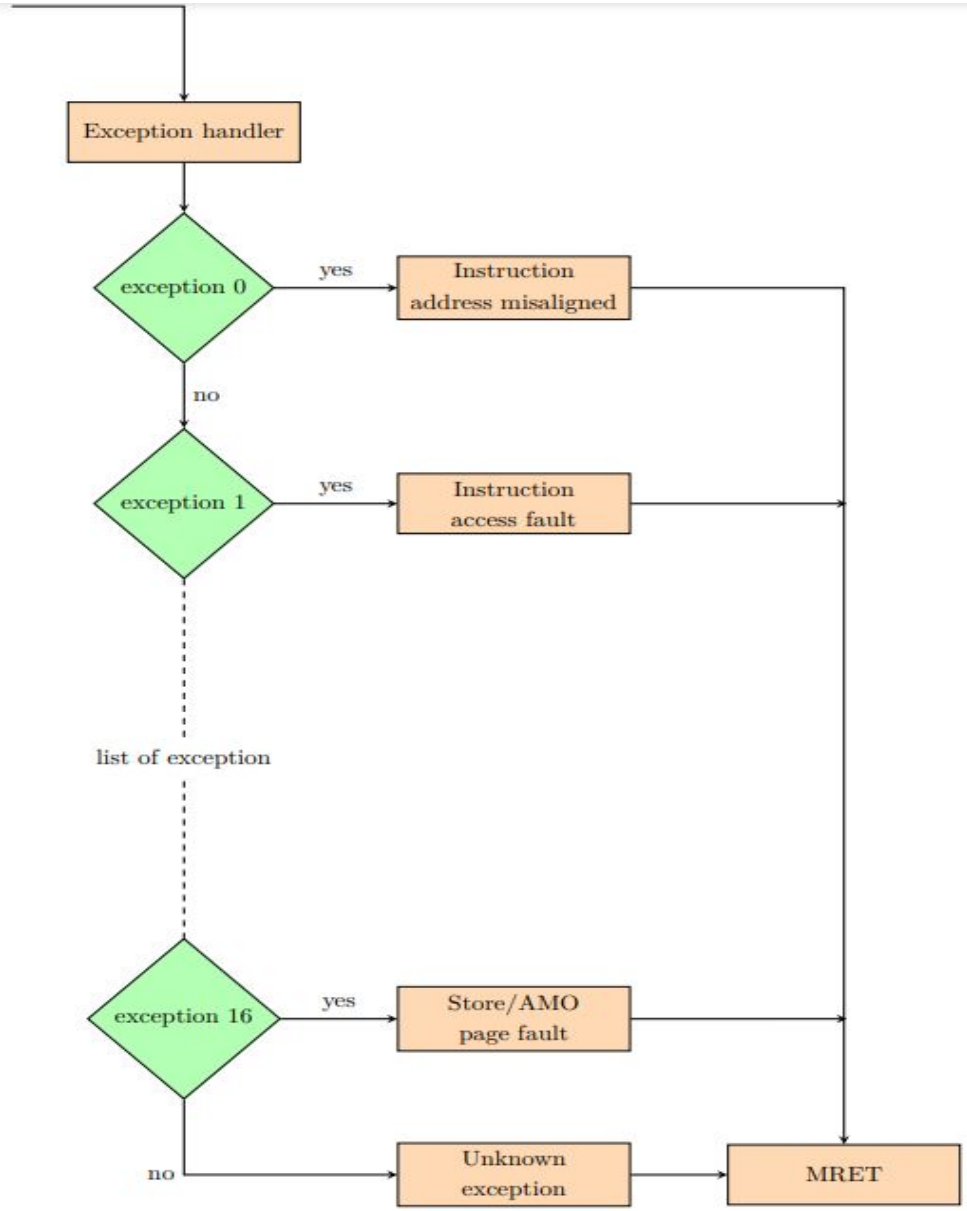
1. The privilege mode is set to Machine Mode.
2. The MIE (Interrupt enable) bit in the status word is set .
3. The MCAUSE register is set to indicate which event has occurred.
4. The MEPC is set to the last instruction that was executing when system Trapped.
5. The PC is set to MTVEC value. Incase of Vectored Traps handling, the PC is set mtvec base address + 4x(mcause)

Interrupt	Exception Code	Description
1	0	User software interrupt
1	1	Supervisor software interrupt
1	2	Hypervisor software interrupt
1	3	Machine software interrupt
1	4	User timer interrupt
1	5	Supervisor timer interrupt
1	6	Hypervisor timer interrupt
1	7	Machine timer interrupt
1	8	User external interrupt
1	9	Supervisor external interrupt
1	10	Hypervisor external interrupt
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
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	Environment call from H-mode
0	11	Environment call from M-mode
0	≥12	*Reserved*

Bits	Field Name	Description
[XLEN-1:6]	Base	Machine Trap Vector Base Address. 64-byte Alignment
[1:0]	Mode	MODE Sets the interrupt processing mode.

mtvec Modes		
Value	Name	Description
0x0	Direct	All Exceptions set PC to mtvec.BASE Requires 4-Byte alignment
0x1	Vectored	Asynchronous interrupts set pc to mtvec.BASE + (4×mcause.EXCCODE) Requires 4-Byte alignment
> 0x01		Reserved


Start
System init
User application running
loop
TRAP_EVENT
Trapped state
priv - M, mepc - pc
mie ⟵ 0
pc - mtvec base addr
Trap_entry
save reg context
save mcause, mepc
in stack
Trap handler
INT
no
Exception handler
yes
Interrupt handler
control transferred
mret
control transferred


Exception handler
exception 0
yes
Instruction address misaligned
no
exception 1
yes
Instruction access fault
list of exception
exception 16
yes
Store/AMO page fault
no
Unknown exception
MRET

- # Exception Handling Registers:

- The exception handling mechanism uses 4/5 registers to know all the information of a Trap,Those registers are CSR registers

1. Machine Status Register (MSTATUS) is used to enable/disable the interrupts. The mstatus register has many more bits. But these are the bits used with respect to a Trap.

- We use MSTATUS register while handling exceptions to read and set the MPP and SPP bits based on the requirement to switch privilege modes
- Example:

```
li t0,0x800
csrrs zero, mstatus, t0        # Setting MPP bits on mstatus register
```

2. Mepc register holds the physical address of the instruction, when exception happened.
3. Mtvec has the base address of the Trap handler,. It is usually referred to as the entry point of the Trap.
4. Mcause has the exception of the Trap.
5. MRET is used to return from a trap handler that is executing in the Machine Mode(MRET may only be executed when running in Machine Mode).