Mensagem do Guto:

*MP Interrupts não tem relação com user-level interrupts.*

*Viste a mensagem que enviei sobre a documentação? Sugiro que use os manuais que estão no moodle.*

*Multi-processor interrupts no RISC-V são uma forma particular de "software interrupts". Trabalharemos em modo "machine", logo, o capítulo 3 do manual da parte privilegiada da ISA é uma boa referência. O "SiFive Interrupt Cookbook" também tem boas explicações sobre o tema.*

links:

<https://riscv.org/wp-content/uploads/2017/05/riscv-privileged-v1.10.pdf> [capitulo 3]

<https://starfivetech.com/uploads/sifive-interrupt-cookbook-v1p2.pdf>

<https://en.wikipedia.org/wiki/Inter-processor_interrupt>

<https://chromitem-soc.readthedocs.io/en/latest/plic.html>

**perguntas:**

plataform-level interrupt = software interrupt?

[gabrielpordeus@gmail.com](mailto:gabrielpordeus@gmail.com) **Não**. Platform-level é a PLIC, software interrupt é um subset do CLINT, que é local.

A: valeuu!!

F: [gabrielpordeus@gmail.com](mailto:gabrielpordeus@gmail.com) mas a PLIC é um interrupt controller global, então ele cuida de interrupts de mais de 1 core.

gp: É outro sistema, que lida com interrupts de váárias coisas, entre eles periféricos (tipo uart), pelo que eu entendi. Pro nosso propósito, é só a clint mesmo. Tanto é que nem tem mais o tema da plic nos seminários.

Segundo o interrupt cookbook:

*The global interrupt controller is termed the Platform Local Interrupt Controller (PLIC). The*

*PLIC provides system level flexibility for dispatching interrupts to a single CPU or multiple CPUs*

*in the system.*

**O que temos que responder:**

1. O que são MP interrupts
2. O que causa elas
3. Por que elas existem/são importantes
4. Como elas funcionam no RISCV
5. Como configurar elas
6. Focar na PLIC, mas mencionar a CLINT (quais interrupçoes ela gera? como sao tratadas?)

**Fatos a se levar em conta:**

**1.3 Privilege Levels**

1. Nós trabalharemos em modo machine (level 3, enconding 11, abreviação M), que é o modo *mais privilegiado* para uma máquina RISC-V

**3.1.7 Privilege and Global Interrupt-Enable Stack in mstatus register (pg 20)**

1. Para habilitar interrupts em *machine mode*, devemos acionar (setar em 1) o **interrupt-enable bit MIE** (machine interrupt enable \_\_eu acho\_\_). Esses bits são usados para garantir atomicidade para os interrupt handlers.
   1. Esses bits estão localizados nos bits menos significativos de *mstatus*
2. Cada modo de privilégio tem tem um stack de dois levels para interrupt-enable bit e modo de privilegio. No caso do modo M, o registrador MPIE mantém o valor do bit interrupt-enable antes do evento que causou a interrupção, e MPP registra o modo de privilégio anterior. No caso do MPP, como M é o modo mais baixo, ele tem 2 bits (um pra modo **U**suario e um pra modo **S**upervisor). Quando um evento é levado do modo S para o modo M, MPIE é setado para o valor que estava em MIE, MIE é setado para 0 e MPP é setado para S
3. Para retornar dos eventos que causaram interrupção, o modo M tem a instrução MRET. Quando ela é executada, que o evento tenha acontecido no modo de privilegio S, então MPP registra S, MIE é setado para MPIE e o modo de privilegio é alterado para S. MPIE é então setado para 1 e MPP é setado para U (ou M caso modo U não seja suportado)

**3.1.14 Machine Interrupt Registers (mip and mie)**

1. O registrador **MIP** contém informações sobre interrupções pendentes que estão habilitadas no registrador **MIE**
2. existem registradores correspondentes em Supervisor-mode (SIP, SIE) e User-mode (UIP, UIE), é possível delegar o tratamento de uma interrupção para outro modo através de **MDELEG**
3. bits de **MSIP**(machine software interrupt pending - gera software interrupts em machine mode) são escritos por acessos aos registradores de controles, que são usados para disparar interrupções inter-processadores em machine-mode
4. interrupções inter-processor em níveis de privilégio mais baixos são feitas através de ABI e SBI calls para o AEE e SEE correspondente, o que pode resultar na escrita, em machine-mode, de um bit em MSIP da hart alvo.
5. Para todos os tipos de interrupções (software, timer, and external), se o modo de privilégio não for suportado os bits de pending e interrupt-enable associados são setados em 0 nos registradores **MIP** e **MIE** respectivos.

**3.1.20 Machine Cause Register (mcause)**

1. quando o evento de interrupção está em modo de máquina(M-MODE) o mcause indica o evento que causou o evento. Se em outro modo, o registrador **NUNCA** sera escrito pela implementacao.
2. O bit mais significativo = 1 indica se o evento foi causado por uma interrupção. o resto de seu conteúdo identifica a última exceção.
3. Não há distinção entre exceções com instrução privilegiadas ou exceções ilegais/incorretas.
4. Nesta seção temos uma tabela de descrições das interrupções (Descrevendo o registrador mcause)

**3.2.3 Wait for Interrupt**

1. WFI (wait for interrup) é uma instrução que provê uma dica para a implementação de que a hart (hardware thread) atual pode ser stalled até que uma interrupção precise dela

-> Como WFI é pra HART, podemos ignorá-la (estamos trabalhando apenas com sofware interrupts)

[gabrielpordeus@gmail.com](mailto:gabrielpordeus@gmail.com) No EPOS, a instruçao WFI é a CPU::halt()

**3.4 Non-Maskable Interrupts (link** [**wikipedia**](https://en.wikipedia.org/wiki/Non-maskable_interrupt)**)**

1. o registrador MEPC é escrito com o endereço da próxima instrução a ser executada depois do NMI ser tratado, e MCAUSE é setado com um valor que indica a causa do NMI.

**Inter-processor interrupts (IPIs)**

<https://marz.utk.edu/my-courses/cosc562/riscv/#plic>

IPIs são tipos especiais de interrupções onde um processador X pode causar uma interrupção em um outro processador Y s X precisar que Y realize uma ação, como por exemplo:

* X realizou uma mudança na TLB e precisa avisar Y pra realizar um flush na TLB e atualiza-la
* X está fazendo o desligamento do sistema, e precisa que Y pare ou termine o que está fazendo para poder desligar.

**Control and Status Registers (CSRs) para interrupts**

Usados para configurar e entregar as interrupts para os respectivos controllers

* **mstatus** - Status register containing interrupt enables for all privilege modes, previous privilege mode, and other privilege level settings.
* **mcause** - Status register which indicates whether an exception or interrupt occurred, along with a code to distinguish details of each type.
* **mie** - Interrupt enable register for local interrupts when using CLINT modes of operation. In CLIC modes, this is hardwired to 0 and interrupt enables are handled using clicintie[i] memory mapped registers.
* **mip** - Interrupt pending register for local interrupts when using CLINT modes of operation. In CLIC modes, this is hardwired to 0 and pending interrupts are handled using clicintip[i] memory mapped registers.
* **mtvec** - Machine Trap Vector register which holds the base address of the interrupt vector table, as well as the interrupt mode configuration (direct or vectored) for CLINT and CLIC controllers. All synchronous exceptions also use mtvec as the base address for exception handling in all CLINT and CLIC modes
* **mideleg** - Machine interrupt delegation register, delega o tratamento de interrupções para outro modo de privilégio.

IMPORTANTE AQUI:

[**3.1.5 CLINT Interrupt Levels, Priorities, and Preemption**](https://starfivetech.com/uploads/sifive-interrupt-cookbook-v1p2.pdf)

For CPU designs that utilize Machine mode only, the CLINT would have the following configuration:

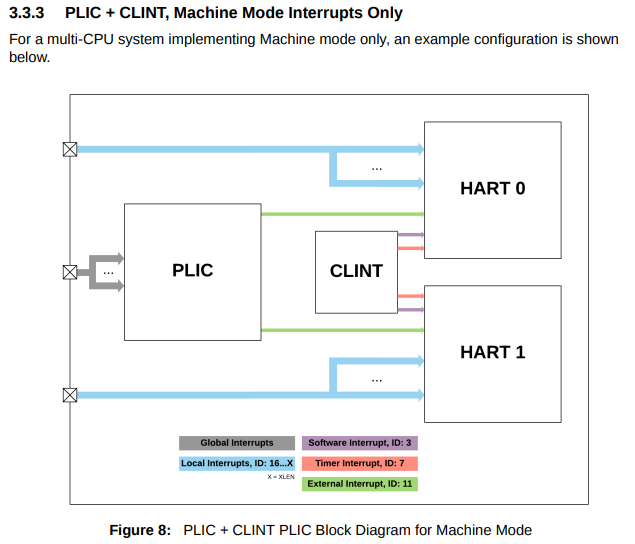
* Software interrupts — Interrupt ID #3.
* Software interrupts are triggered by writing the memory mapped interrupt pending register msip for a particular CPU. **In a multi-CPU system, other CPUs are able to write msip to trigger a software interrupt on any other CPU in the system. This allows for efficient inter-processor communication.**

***NESSE MESMO LINK ACIMA, O 3.3 (PLIC) TAMBÉM É IMPORTANTE***.

The PLIC is used to manage all global interrupts and route them to one or many CPUs in the

system. It is possible for the PLIC to route a single interrupt source to multiple CPUs. Upon

entry to the PLIC handler, a CPU reads the claim register to acquire the interrupt ID. A successful claim will atomically clear the pending bit in the PLIC interrupt pending register, signaling to the system that the interrupt is being serviced.



## CLINT - Core Local Interrupter

Tem um esquema fixo de prioridades que implementa interrupções de **Software**, Timer e Externas.

To configure CLINT modes, write mtvec.mode field, which is bit[0] of mtvec CSR. For direct

mode, mtvec.mode=0, and for vectored mode mtvec.mode=1. Direct mode is the default reset

value, while mtvec.base holds the base address for interrupts and exceptions in both modes

# MultiProcessor interrupts

1. O que são interrupts
   1. Qual sua importância
   2. Exemplo de um interrupt (que não seja de usuário, ou seja, de IO)
2. Como os interrupts acontecem em um multi-processador
   1. Como um processador pode pedir pra interromper outro?
   2. Pra que isso pode ser feito?
      1. Importância (possibilitar compartilhamento de memória, por exemplo)
      2. Exemplo
3. MP Interrupts em RISC V
   1. Como habilitar (falar de machine mode)
   2. Como emitir
   3. Como é gerenciado
      1. PLIC
      2. CLINT
   4. Como retornar ao programa

== == == == == == == == == == == == == == == == == == == == == == == == == == == ==

# O que são interrupts

## Qual sua importância

Interrupts são importantes por diversos motivos, entre eles podemos citar a transferência de poder do sistema para o usuário (em um input por exemplo), comunicação com o kernel e o tratamento de erros e exceções por meio de system calls

## Exemplo de um interrupt (que não seja de usuário, ou seja, de IO)

# Como os interrupts acontecem em um multi-processador

## Como um processador pode pedir pra interromper outro?

## Pra que isso pode ser feito?

## Importância (possibilitar compartilhamento de memória, por exemplo)

## Exemplo

# MP Interrupts em RISC V

## Como habilitar (falar de machine mode)

## Como emitir

## Como é gerenciado

## PLIC

## CLINT

## Como retornar ao programa

A PLIC (Platform Local Interrupt Controller) é usada para gerenciar todos os interrupts globais e direcionar eles para uma ou mais CPUs no sistema, garantindo mais flexibilidade no nível de sistema.

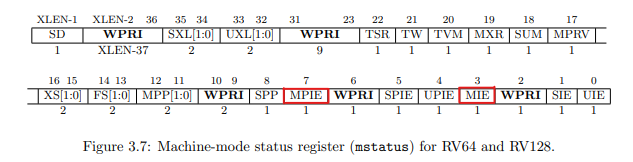
Ela consegue direcionar uma fonte de interrupt para multiplos CPUs. Quando a PLIC recebe uma nova entrada, uma CPU lê o registrador chamado para adquirir o ID do interrupt, e, caso tenha adquirido com sucesso, ela limpa atomicamente o bit de *pendente* na PLIC (IPR - interrupt pending register), sinalizando ao sistema que o interrupt está sendo tratado.

| Para habilitar interrupts em machine mode, devemos acionar (setar em 1) o

| interrupt-enable bit MIE (machine interrupt enable). Esses bits são usados

| para garantir atomicidade para os interrupt handlers e **estão localizados nos bits**

| **menos significativos de mstatus.**



Além disso, o bit MIE é usado para garantir atomicidade para os interrupt handlers, pois ele pode ser setado com uma instrução CSR. (O mesmo é valido para SIE e UIE)

Para as explicações abaixo, assumimos execuções em modo *machine* (M)

Cada modo de privilégio tem tem um stack de dois levels para interrupt-enable bit e modo de privilegio. No caso do modo M, o registrador MPIE mantém o valor do bit interrupt-enable antes do evento que causou a interrupção, e MPP registra o modo de privilégio anterior. No caso do MPP, como M é o modo mais baixo, ele tem 2 bits (um pra modo Usuario e um pra modo Supervisor). Quando um evento é levado do modo S para o modo M, MPIE é setado para o valor que estava em MIE, MIE é setado para 0 e MPP é setado para S

Para retornar dos eventos que causaram interrupção, o modo M tem a instrução MRET. Quando ela é executada, assumindo que o evento tenha acontecido no modo de privilegio S, então MPP registra S, MIE é setado para MPIE e o modo de privilegio é alterado para S. MPIE é então setado para 1 e MPP é setado para U (ou M caso modo U não seja suportado).

## Dúvidas atendimento 30/05

Qual seria a melhor tradução pra *traps*? - pode ser traduzido como interrupção mesmo

Explicação de:

*To support nested traps, each privilege mode x has a two-level stack of interrupt-enable bits and privilege modes. xPIE holds the value of the interrupt-enable bit active prior to the trap, and xPP holds the previous privilege mode. The xPP fields can only hold privilege modes up to x, so MPP is two bits wide, SPP is one bit wide, and UPP is implicitly zero. When a trap is taken from privilege mode y into privilege mode x, xPIE is set to the value of x IE; x IE is set to 0; and xPP is set to y.*

O seminário deve abordar: (dúvida)

1. Sobre interrupçoes (o que são, como acontecem, pra que servem, exemplos de interrupções em cores diferentes)

**R: como um core gera interrupção em outro, o que tem que ser escrito em assembly pra gerar essa interrupção, o que tem que ser escrito pra tratar essa interrupção**

1. Como elas acontecem em RISC V
2. Como elas são tratadas (PLIC + CLINT [mais por cima]) -> (que interrupções geram e como são tratadas)

Isso já seria completo o suficiente?

o que sao interrupções e pra que servem em multicores

qual assembly precisa ser escrito pra gerar uma interruçao

qual assembly precisa ser escrito pra tratar uma interrupcao

vetores de interrupcao

CLINT -> regs dizendo se é interrupção de software e etc

* Regs do CLINT estao mapeados em mem + offset -> pra gerar uma interrupçao em um core a gente precisa escrever num bit especifico
* WAIT FOR INTERRUPT num core pra esperar uma interrupção

nosso demo:

outros cores sobem e ficam em WFI

um core manda interrupção pra outros cores (com delay) - esses cores acordam, imprimem algo e voltam a dormir.

Quais regs do RISC V sao pertinentes pra interrupcao

vetor de interrupçao

falar que tem como delegar pra outro nivel (MDELEG)

Referencias

<https://www.techopedia.com/definition/22195/software-interrupt#:~:text=A%20software%20interrupt%20is%20invoked,during%20error%20or%20exception%20handling>.

Roteiro novo:

1. O que são interrupçoes (não ser muito generalista, apenas uma introdução), qual importância tem (dar um exemplo, sem envolver IO, como por exemplo pedido de flush de memoria em outro core)
2. Quais registradores em RISC V são pertinentes
3. Como gerar uma interrupçao - em assembly mesmo
   1. Em qual registrador devemos escrever, o que devemos escrever la, o que significam os regs e os offsets
4. Como tratar uma interrupção
   1. Aonde/como o outro core recebe a interrupção
   2. O que são e pra que server os vetores de interrupções
5. O que é WFI (wait for interruption)