Interruptions et exceptions

2017 – 2018

Florent Gluck – Florent.Gluck@hesge.ch

Version 0.5

h e p i a
Haute école du paysage, d'ingénierie et d'architecture de Genève

Hes·so // GENÈVE

Interruptions

- On peut distinguer 3 types d'**interruptions** :
  - **Interruptions logicielles**
  - **Interruptions matérielles**
  - **Exceptions processeur**

- L'architecture IA-32 supporte jusqu'à 256 interruptions (numérotées de 0 à 255).

- Intel **réserve** les 32 premiers n° d'interruptions (0 à 31) pour les exceptions processeur.

Interruptions logicielles (1)

- Une interruption logicielle est exécutée avec l'instruction `INT`.
- L'instruction `INT 0x10` exécute l'interruption 0x10.
- Au moment de l'appel à l'instruction INT, le pointeur d'instruction saute à l'adresse du code correspondant au numéro de l'interruption logicielle spécifiée.
- La table des descripteurs d'interruption (IDT) défini, pour chaque interruption (logicielle, matérielle, exception), l'adresse du code à exécuter pour chaque numéro d'interruption.

Interruptions logicielles (2)

- Les interruptions logicielles sont **toujours synchrones** vis-à-vis de l'exécution du programme :
  - Cela signifie qu'au moment où l'instruction `INT` est exécutée, l'interruption logicielle dont le numéro est spécifié en argument est exécutée. Le comportement est similaire à l'exécution d'une fonction.

- Au contraire, les interruptions matérielles sont **asynchrones** car elles proviennent du matériel et peuvent donc être déclenchées à n'importe quel moment.

Interruptions matérielles

- Typiquement générées par des périphériques (disques, clavier, etc.).
- Il existe deux types d'interruptions matérielles :
  - Non Maskable Interrupts (NMI)
  - Interrupt Requests (IRQ)
- Une NMI implique un problème matériel (mémoire défectueuse, erreur de bus, etc.). Les NMI ne **peuvent pas** être ignorées ou masquées. But : empêcher la machine de fonctionner afin d'éviter des pertes de données.
- Les IRQ sont générées par des périphériques. L'instruction `CLI` permet de les masquer et l'instruction `STI` de les démasquer.
- Contrairement aux interruptions logicielles, les NMI ou IRQ sont **asynchrones**. Elle peuvent être déclenchées à **n'importe quel moment** (n'importe quand/où).

Exceptions processeur

- Les exceptions processeur sont générées par le CPU lui-même.
- Elles sont similaires à une interruption logicielle, donc **sychrones**.
- Quand surviennent-elles ?
  - Lorsque le CPU n'est pas capable de gérer une erreur causée par le code en exécution (logiciel).
  - Exemples :
    - division par zéro
    - erreur de protection générale (p.ex : accès mémoire invalide)
    - Instruction non supportée par le matériel (instruction SSE3 alors que seulement SSE2 disponible, etc.)
    - etc.

Exceptions processeur (IA-32)

Vector	Mne-monic	Description	Type	Error Code	Source
0	#DE	Divide Error	Fault	No	DIV and IDIV instructions.
1	#DB	RESERVED	Fault/ Trap	No	For Intel use only.
2	—	NMI Interrupt	Interrupt	No	Nonmaskable external interrupt.
3	#BP	Breakpoint	Trap	No	INT 3 instruction.
4	#OF	Overflow	Trap	No	INTO instruction.
5	#BR	BOUND Range Exceeded	Fault	No	BOUND instruction.
6	#UD	Invalid Opcode (Undefined Opcode)	Fault	No	UD2 instruction or reserved opcode.[1]
7	#NM	Device Not Available (No Math Coprocessor)	Fault	No	Floating-point or WAIT/FWAIT instruction.
8	#DF	Double Fault	Abort	Yes (zero)	Any instruction that can generate an exception, an NMI, or an INTR.
9		Coprocessor Segment Overrun (reserved)	Fault	No	Floating-point instruction.[2]
10	#TS	Invalid TSS	Fault	Yes	Task switch or TSS access.
11	#NP	Segment Not Present	Fault	Yes	Loading segment registers or accessing system segments.
12	#SS	Stack-Segment Fault	Fault	Yes	Stack operations and SS register loads.
13	#GP	General Protection	Fault	Yes	Any memory reference and other protection checks.
14	#PF	Page Fault	Fault	Yes	Any memory reference.
15	—	(Intel reserved. Do not use.)		No	
16	#MF	x87 FPU Floating-Point Error (Math Fault)	Fault	No	x87 FPU floating-point or WAIT/FWAIT instruction.
17	#AC	Alignment Check	Fault	Yes (Zero)	Any data reference in memory.[3]
18	#MC	Machine Check	Abort	No	Error codes (if any) and source are model dependent.[4]
19	#XM	SIMD Floating-Point Exception	Fault	No	SSE/SSE2/SSE3 floating-point instructions[5]
20	#VE	Virtualization Exception	Fault	No	EPT violations[6]
21-31	—	Intel reserved. Do not use.			
32-255	—	User Defined (Non-reserved) Interrupts	Interrupt		External interrupt or INT *n* instruction.

Requête d'interruption (1)

- Une requête d'interruption (IRQ) est une interruption générée par le matériel, typiquement un périphérique.
- Des IRQ sont générées par tout type de périphérique : disque dur, carte réseau, carte son, clavier, souris, interface USB, etc.
- En général un périphérique génère une IRQ lorsque des données sont prêtes à être lues ou lorsqu'une commande est terminée (ex: écriture d'un buffer sur disque, pression d'une touche, etc.).
- En d'autre terme, **une IRQ est générée lorsqu'un périphérique nécessite l'attention du CPU**.

Requête d'interruption (2)

- Lorsqu'un périphérique signale une IRQ, une interruption matérielle est générée :
  - Le CPU est interrompu de manière **asynchrone** et exécute la routine d'interruption (ISR) correspondant à l'IRQ.

- A toute IRQ correspond une interruption matérielle :
  - Un mapping existe donc entre IRQ et interruption
  - Ce mapping est réalisé par le contrôleur d'interruption (PIC)

- Le CPU exécute le code de l'ISR ; exemple : lire un caractère au clavier, lire un paquet sur une carte réseau, etc.

- Une fois la routine ISR terminée, le CPU doit notifier le PIC en envoyant une commande *End Of Interrupt* (EOI)
  - Sinon, l'interruption ne sera jamais plus déclenchée.

Table des IRQ et mapping par défaut

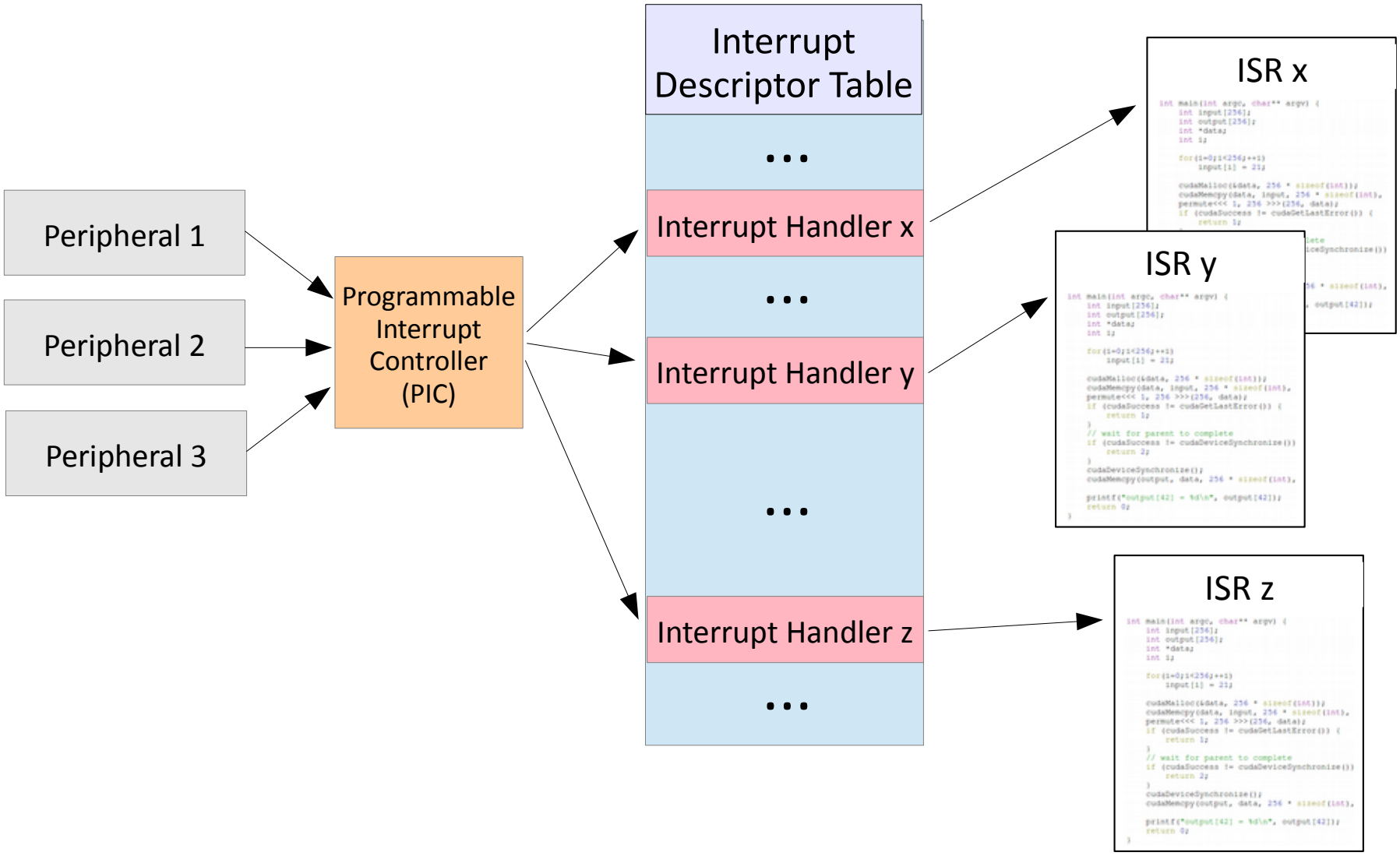
IRQ	Description	Interruption
0	System timer (PIT)	0x08
1	Keyboard	0x09
2	Redirected to slave PIC	0x0A
3	Serial port (COM2/COM4)	0x0B
4	Serial port (COM1/COM3)	0x0C
5	Sound card	0x0D
6	Floppy disk controller	0x0E
7	Parallel port	0x0F
8	Real time clock	0x70
9	Redirected to IRQ2	0x71
10	Reserved	0x72
11	Reserved	0x73
12	PS/2 mouse	0x74
13	Math coprocessor	0x75
14	Hard disk controller	0x76
15	Reserved	0x77

Interrupt Descriptor Table

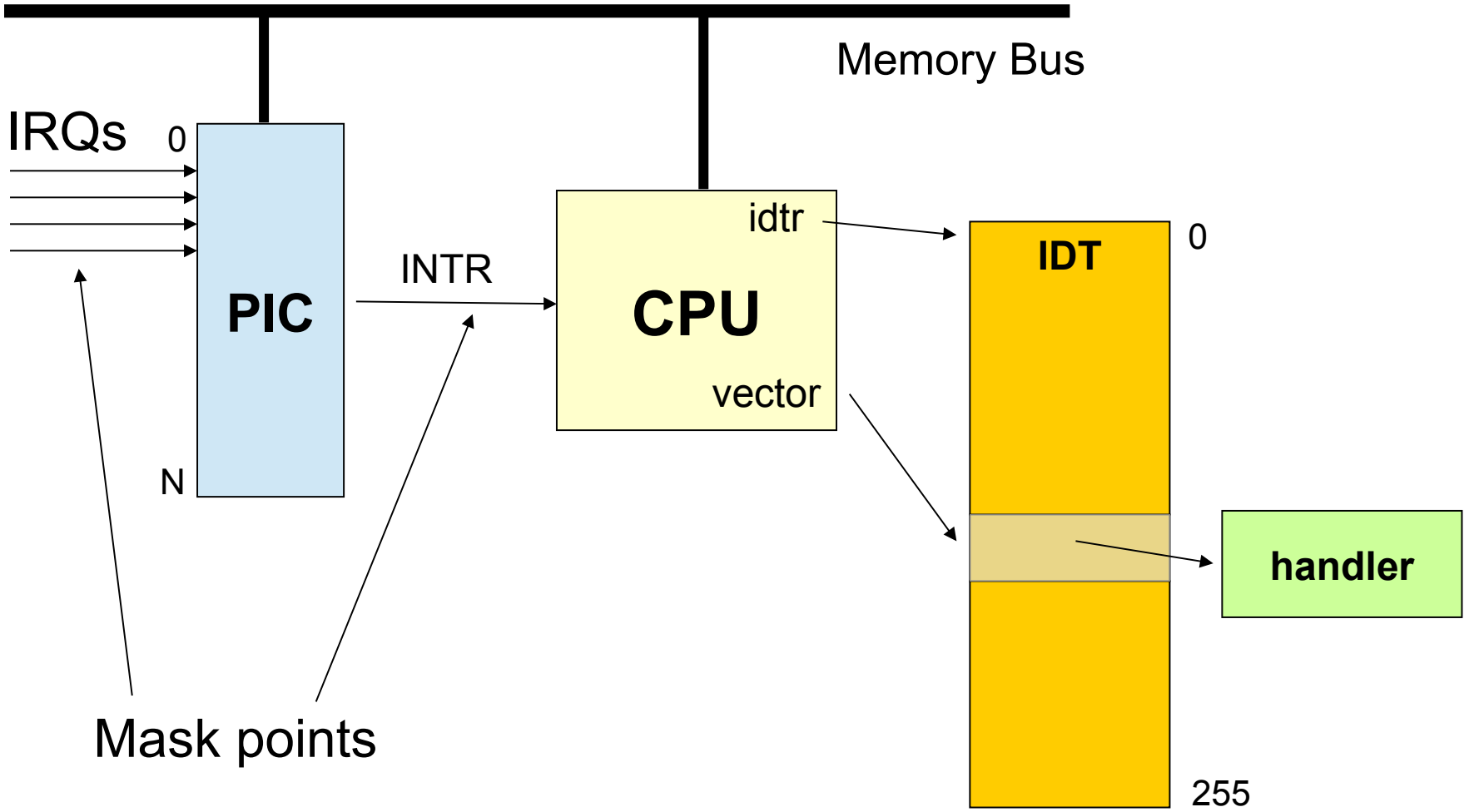
- Le CPU utilise une table des descripteurs d'interruption (Interrupt Descriptor Table – IDT) pour localiser la routine d'interruption à exécuter lorsque :
  - Une interruption logicielle est appelée (instruction INT) ;
  - Une interruption matérielle est déclenchée (via une IRQ) ;
  - Une exception processeur est levée.

- Tout comme la GDT, l'IDT réside en mémoire vive et doit être initialisée et gérée par le système d'exploitation.
  - Pourquoi ? Qu'en est-il si ce n'est pas le cas ?

IRQ : vue globale



IRQ : du matériel au logiciel

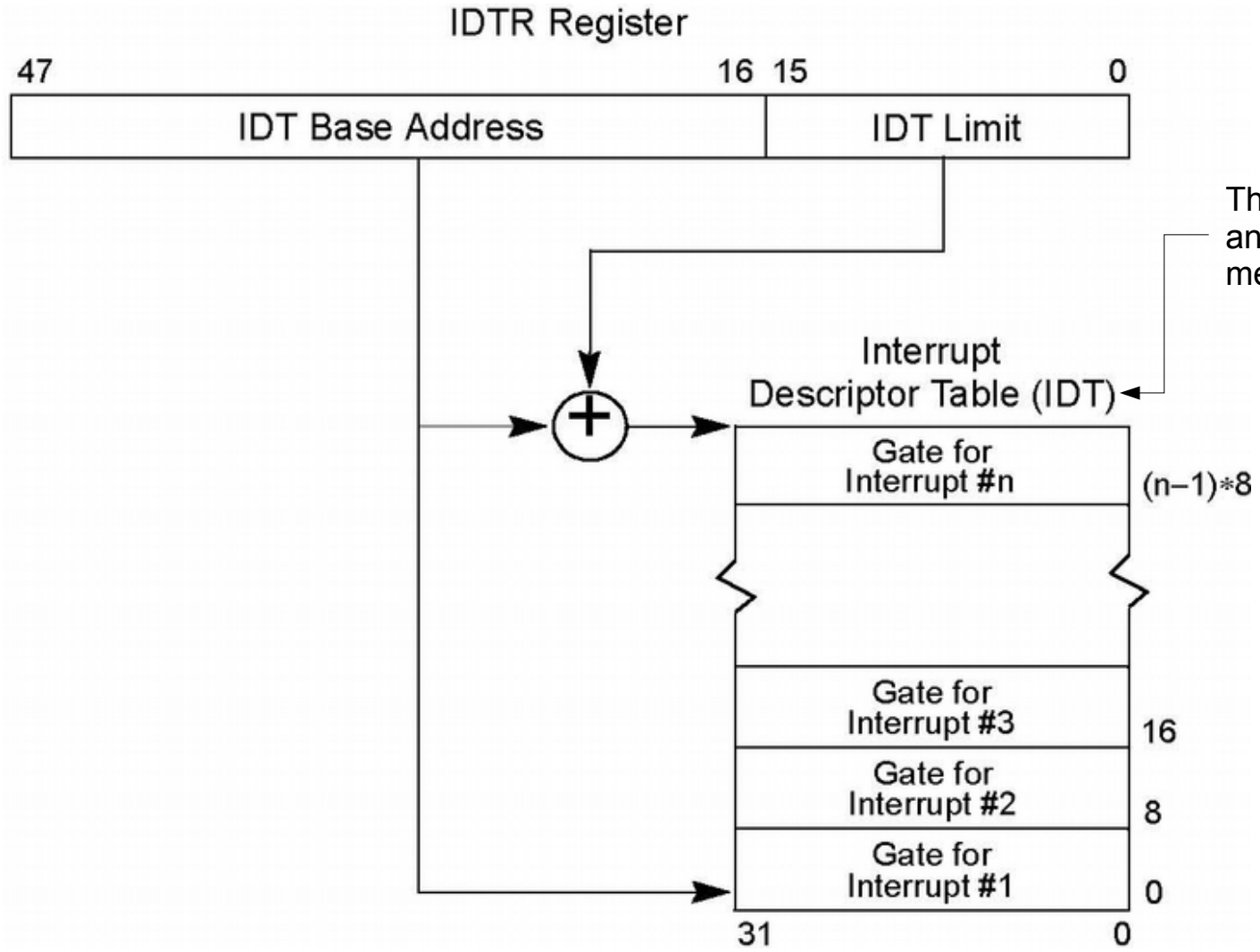


Descripteurs d'interruption

- Similairement à la GDT, la table IDT contient des descripteurs ; il s'agit de descripteurs d'interruption de 64-bits aussi appelé *gates.*

- Un descripteur d'interruption contient :
  - **Un offset** indiquant l'adresse de la routine à exécuter (ISR).
  - **Un selecteur de segment** indiquant le segment dans lequel se trouve le code de l'ISR ; cela permet au CPU de donner le contrôle au kernel grâce à une interruption générée depuis un autre niveau de privilège, typiquement une application s'exécutant en ring 3 (mode utilisateur).
  - **Un niveau de privilège** indiquant le niveau de privilège requis pour exécuter l'ISR.

IDT et registre IDTR (1)



IDT et registre IDTR (2)

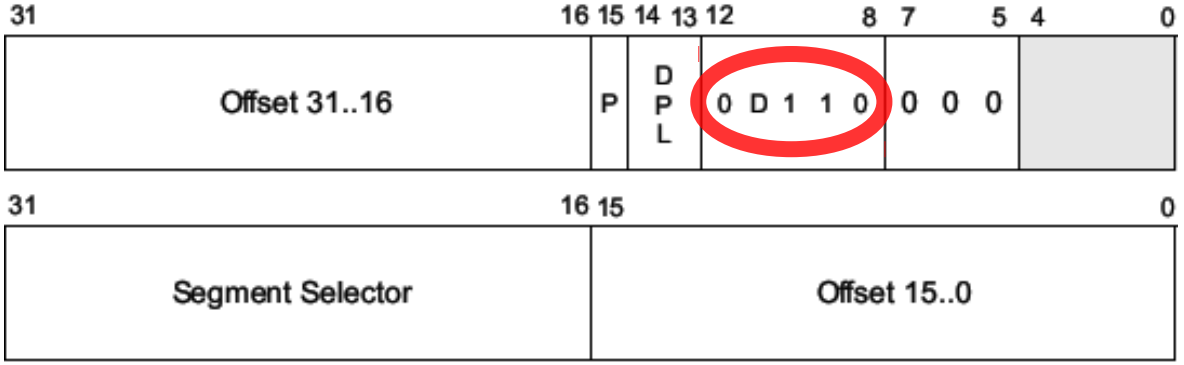
- De manière similaire à la GDT, l'IDT est localisée par le CPU grâce au registre IDTR. Ce registre de 48 bits contient :
  - Adresse de base de l'IDT (32 bits).
  - Limite de l'IDT (16 bits) ; cette limite, tout comme pour la GDT, indique la taille en bytes – 1 de l'IDT.
- L'instruction assembleur `lidt` permet de charger l'IDT dans le registre `idtr` (similairement à `lgdt` et `lgdtr` pour la GDT) afin qu'elle soit prise en compte par le CPU.
- L'instruction `lidt` prend comme unique argument l'adresse d'une structure de 48 bits qui spécifie l'adresse de base de l'IDT et sa taille.
- Exemple de chargement :
  - `lidt [eax]` où eax pointe sure la structure de 48 bits

Interrupt gates et trap gates

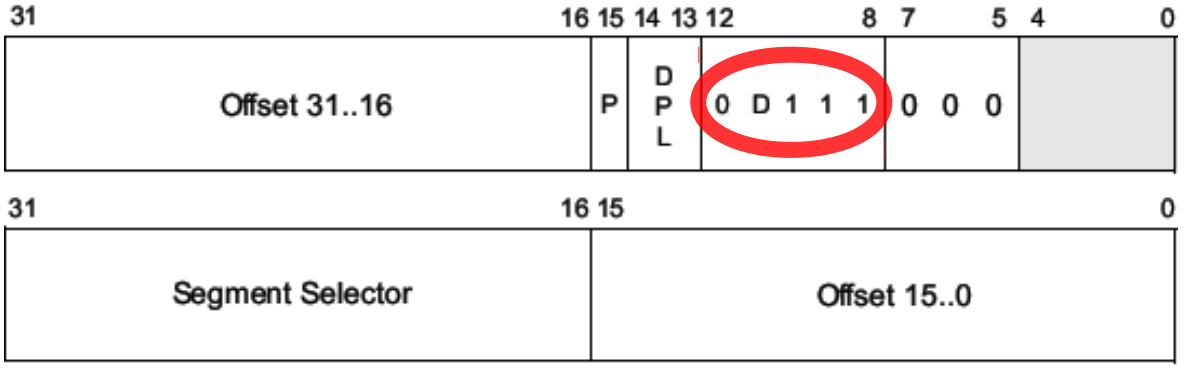
- L'IDT peut contenir différents types de descripteurs d'interruption.
- Nous présentons ici deux types de descripteurs :
  - Interrupt Gate
  - Trap Gate
- La différence entre un Interrupt Gate et un Trap Gate est uniquement le comportement du CPU durant l'exécution de la routine d'interruption (ISR).
- Dans le cas d'un Interrupt Gate, **les interruptions sont masquées** durant l'exécution de l'ISR.
- Dans le cas d'un Trap Gate, les interruptions **ne sont pas masquées** durant l'exécution de l'ISR (le Flag IF du registre EFLAGS reste inchangé).

Interrupt Descriptors

Interrupt Gate



Trap Gate



DPL	Descriptor Privilege Level
Offset	Offset to procedure entry point
P	Segment Present flag
Selector	Segment Selector for destination code segment
D	Size of gate: 1 = 32 bits; 0 = 16 bits

Reserved

Interrupt/trap gate : exécution

- Que se passe-t-il lorsque le CPU exécute l'instruction `INT 60` ?
- Il s'agit d'une interruption logicielle :

  1) Le CPU localise le descripteur à l'index 60 de l'IDT.

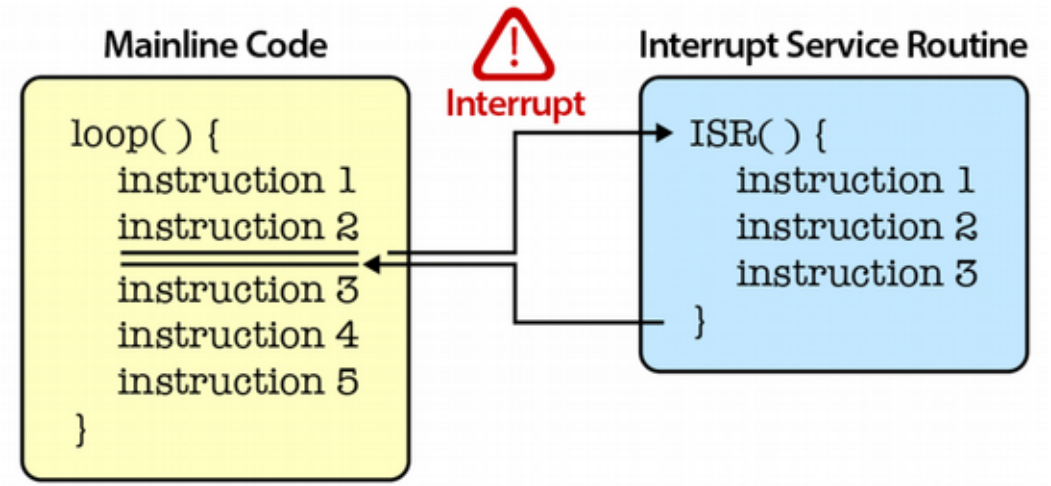
  2) Le CPU lit le descripteur et extrait le **segment selector** et l'adresse **offset** de la routine d'interruption à appeler (ISR).

  3) Le CPU place les informations de retour sur la pile (`EFLAGS`, `CS`, `EIP`) ce qui permettra de revenir au code appelant grâce à l'instruction de retour d'interruption `IRET`.

  4) Le CPU saute à la routine d'interruption se trouvant à l'adresse **offset** dans le segment référencé par **segment selector**.

Contexte

- La nature asynchrone des interruptions implique que le processeur peut-être interrompu à **n'importe quel moment** :



- La routine d'interruption (Interrupt Service Routine) doit donc **sauvegarder** le contexte courant du processeur et le **rétablir** à la sortie de la routine.

Routine d'interruption (1)

- Toute routine d'interruption (ISR) doit sauvegarder le contexte du CPU et mettre à jour le registre DS afin de pouvoir accéder aux bonnes données.

- **Toute routine d'interruption doit être le plus court possible** car pendant toute la durée du traitement de celle-ci, toutes les autres interruptions matérielles sont désactivées !

Routine d'interruption (2)

- Dans le cas des exceptions processeur, un code d'erreur peut-être (ou pas, selon l 'exception) déposé sur la pile.

- En général, les routines d'interruption se composent de deux parties :
  - Une partie bas niveau, en assembleur, qui s'occupe de :
    - Sauvegarder (début) et restaurer (fin) le contexte.
    - Appeler le gestionnaire d'interruption haut niveau en C.
  - Une partie haut niveau, en C, qui implémente le gestionnaire d'interruption.