TD 3.1
Archi 1

Architecture des ordinateurs.

Les interruptions.

- Une interruption est une exception, généralement de cause externe, et qui nécessite d'interrompre l'exécution du programme courant pour être traitée.

- Quelques causes d'interruptions classiques :
  - → interaction clavier / souris
  - → Timer
  - → Arrivée d'un paquet réseau

- Elles sont gérées par un gestionnaire d'interruptions, c'est-à-dire un programme qui va traiter les différentes actions à effectuer liées à l'interruption.

- En CRAPS, lorsqu'une interruption survient, les actions suivantes sont effectuées :
  - → On termine l'exécution de l'instruction courante
  - → **push %r30** : sauvegarde de l'adresse de l'instruction courante
  - → **push %r25** (psr : Program Status Register) : Sauvegarde des flags N, Z, V et C qui sont stockés dans le registre %r25
  - → **ba 0x1** : On se branche à l'adresse du gestionnaire d'interruptions

- Cette dernière opération a un impact sur la manière d'écrire les programmes avec interruption en CRAPS.

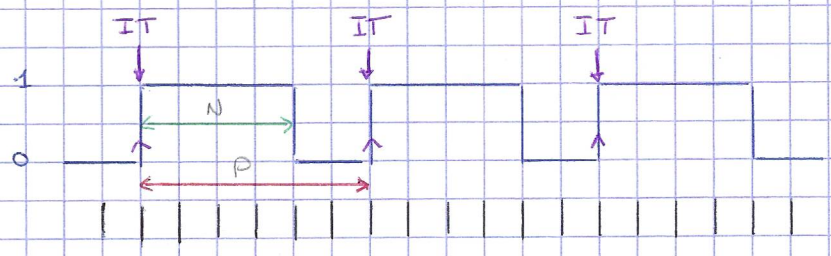
- Les programmes auront la structure suivante :

```
handler : ba Main
          ____
          ____
          reti

Main    : set 0x200, %r29 // Init du pointeur de pile
          ____
          ____
```

- Un gestionnaire d'interruptions s'achève nécessairement par l'instruction **reti** qui réalise les opérations suivantes :
  - → **pop %r25** : On rétablit les flags.
  - → **pop %r30** : On retourne à l'instruction interrompue.

- Dans CRAPS, il existe deux sources d'interruptions de même niveau, traitées par le même gestionnaire d'interruptions :
  - → Appui sur le bouton poussoir btn3 situé en bas à droite de la carte.
  - → A chaque front montant d'un signal périodique généré par le module PWM (Pulse-width Modulation)



- exemple de signal généré par le PWM.
- cycles d'horloge du PWM (195 312 Hz)

- P : période du signal en nombre de cycles d'horloge du PWM
- N : durée pendant laquelle le signal est à 1 en nombre de cycles du PWM
- P et N peuvent être écrits respectivement aux adresses 0xC000 0000 et 0xC000 0001.

Exercice 1 : Ecrire un programme qui incrémente un compteur à chaque appui sur btn 3, et qui affiche la valeur de ce compteur.

On pourra réaliser la fonction : Afficher // IN, %r1 valeur à afficher
// IN, %r2 masque d'activation

```
        ba main
handler : push %r2
          ld [0x100], %r2
          add %r2, 1, %r2
          st %r2, [0x100]
          pop %r2
          reti
main    : set 0x2006, %r29
          st 0b1111, %r2          // Activation des 4 afficheurs
loop    : ld [0x100], %r1         // Début de la boucle d'Affichage, et lecture du
          call Afficher                compteur en mémoire
          ba loop
```

st %ri, [0x100] : écriture du contenu de %ri à l'adresse 0x100 en mémoire

ld [0x100], %ri : écriture du mot-mémoire d'adresse 0x100 dans le registre %ri

Exercice 2 : Ecrire un programme qui incrémente un compteur toutes les secondes et qui affiche la valeur de ce compteur.

```
        ba main
handler : push %r2
          ld [0x100], %r2
          add %r2, 1, %r2
          st %r2, [0x100]
          pop %r2
          reti
main    : set 0x200, %r29
          st 0b1111, %r2          // Activation des 4 afficheurs
          set 0xC000 0000, %r3    // Adresse des paramètres du PWM
          set 195312, %r4         // %r4 = P = 195312
          st %r4, [%r3]
          st %r20, [%r3+1]        // N = 1 (%r20)
loop    : ld [0x100], %r1
          call Afficher
          ba loop
```

TD 3.2.

Archi 1

Exercice 3: Ecrire un programme qui affiche alternativement Pile (0x AAAA) et FACE (0x FACE) et qui se met en pause pendant 2 sec lors de l'appui sur btn3.

```
        ba main.
handler: push %r28
         call delay 1sec
         pop %r28
         push %r28
         call delay 1sec
         pop %r28.
         ba main.

main:  set 0x200, %r29.
       set 0b1111, %r2  // Activation des 4 afficheurs.
loop:  set 0xAAAA, %r1. // Affiche Pile
       call Afficher
       set 0xFACE, %r1 // Affiche Face.
       call Afficher
       ba loop.
```