Tricks

Multiplication avec shift et log

Quand on veut calculr $a \cdot b$, si on dispose du $\log_a$ on peut faire $b \ll \log_a$.

Charger un mot au milieu

Toutes les adresses doivent être alignées sur la taille des données qu'elles manipulent (donc quand on fait un `lw` ça doit être un multiple de 4). Mais RISCV va modifier le code pour tomber sur le mot juste avant si on utilise un `lw` qui n'est pas un multiple de 4. `lw_ecrit = 4k + r`, donc `lw_corrige = 4k`.

Charger un immédiat avec 32 bits

`li` supporte le chargement d'un immediate de 32 bits (tandis que toutes les autres opérations comme `addi`, la constante devant `lw`, etc.) ne supportent que des 12 bits.

Convention d'appel

Même si on sait que la fonction `b` que notre fonction `a` va appeler ne modifie pas les saved registers, la fonction appelante doit **TOUJOURS** utiliser et sauvegarder les saved registers

```
addi sp, sp, -12 # mettre à jour le stack pointer (on le fait remonter dans la
mémoire)
sw ra, 0(sp)
sw s1 4(sp)
sw s2, 8(sp) # 8... + 4 = 12 bytes libérés utilisés

# blabla

lw ra, 0(sp)
lw s1, 4(sp)
lw s2, 8(sp)
addi sp, sp, 12
```