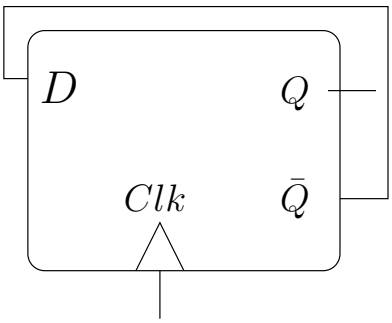
Devoir Surveillé n° 2

Durée : 1h30 — Sans document

Ce sujet comporte 1 page recto-verso

1 Circuit séquentiel et horloge

Le circuit ci-contre est un bistable D sur lequel on a relié la sortie "inversée" $\bar{Q}$ à l'entrée D. On suppose qu'initialement, la sortie Q vaut 0.



Q1 En utilisant un chronogramme s'étalant sur au moins 4 cycles d'horloge, tracez l'évolution de la sortie Q en regard de la valeur du signal d'horloge. Comparez les 2 signaux. Que constatez-vous ?

Q2 Déduisez-en une façon de réaliser un circuit "diviseur de fréquence" qui accepte en entrée un signal d'horloge ainsi qu'un bit de commande (0 = normal, 1 = slow) et qui fournit en sortie soit le signal d'horloge inchangé (mode normal) soit un signal d'horloge de période deux fois plus longue (mode slow).

Q3 Proposez un circuit composé de deux bistables D qui permette de diviser la fréquence d'horloge par quatre. Tracez le chronogramme s'étalant sur au moins 8 cycles d'horloge.

2 Y86 : fonction swap

Soit la fonction C `swap` ci-dessous qui inverse les valeurs des deux variables qui sont passées en paramètres en tant que pointeur :

```
void swap(int *x, int *y) {
int temp = *x;
*x = *y;
*y = temp;
}
```

Soit l'extrait de code Y86 suivant :

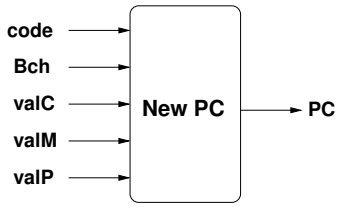
```
root: irmovl 0x200, %esp
      ...
      halt
swap: ...
      ret
.pos 0x100
a:  .long  4
b:  .long  5
```

Q1 Complétez le début du programme (la fonction `root`) afin d'appeler la fonction swap en utilisant les variables a et b.

Q2 Proposez une implémentation en Y86 de la fonction `swap` basée sur le code C fourni.

3 Version séquentielle du simulateur Y86

La figure ci-contre montre les entrées du circuit combinatoire « `New PC` » chargé de calculer l'adresse de la prochaine instruction que le processeur devra charger (l'entrée `Bch` désigne la valeur « *Branch Flag* »). Le code HCL « partiel » décrivant ce circuit est donné sous le schéma.



Q1 Expliquez le code HCL fourni.

Q2 Complétez ce code pour traiter les quelques cas non gérés actuellement. Pour chaque ligne ajoutée, expliquez précisément le choix de la valeur résultat.

```
new_pc = [
  code == Jxx && Bch : valC ;
  ...
  1 : valP ;
]
```

4 Version pipelinée du simulateur Y86

On considère la version pipelinée du processeur Y86 à 5 étages (*Fetch*, *Decode*, *Execute*, *Memory*, *Write Back*) telle que vue en cours.
On souhaite examiner le fonctionnement du processeur lors de l'exécution de la séquence d'instructions ci-contre.

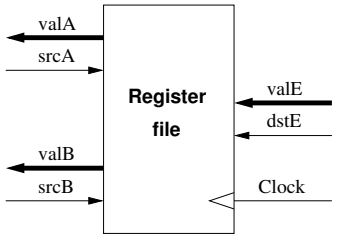
```
irmovl var1, %ebx
mrmovl 4(%ebx), %eax
isubl 2,%eax
iaddl 4, %ebx
rmmovl %eax, (%ebx)
```

Q1 En supposant une version naïve du processeur se limitant à l'insertion automatique de *bulles* pour conserver une exécution correcte, dessinez un chronogramme montrant la progression des instructions à l'intérieur du pipeline pour chaque cycle.

Q2 En supposant maintenant une version du processeur capable de propager des valeurs des étages *E*, *M* ou *WB* vers la fin de l'étage *D*, redessinez le chronogramme. Y a-t-il encore apparition de *bulles* ? Expliquez.

5 Les registres de Y86

Voici un schéma ci-contre qui présente une version légèrement simplifiée du fichier de registres de Y86 qui doit gérer 8 registres. Nous rappelons que les flèches en gras (`valA`, `valB` et `valE`) indiquent des entrées ou sorties de données sur 32 bits. Les autres flèches indiquent des entrées de données sur 3 bits. L'entrée `srcA` (resp. `srcB`) code le nom du registre dont le contenu devra être envoyé sur la sortie `valA` (resp. `valB`). L'entrée `dstE` code le nom du registre dans lequel il faudra écrire les données qui arrivent sur `valE`. Dans cette version simplifiée, on suppose que l'on écrit toujours valE dans un registre.



Les questions ci-dessous sont indépendantes.

Q1 Pourquoi les entrées `srcA`, `srcB` et `dstE` sont-elles simplement codées sur 3 bits ?

Q2 Proposez une implémentation simple d'un registre 4 bits. On rappelle qu'un registre est un circuit séquentiel qui possède une entrée et une sortie de données sur n bits ($n = 4$ pour cette question), et une entrée supplémentaire *Load* qui détermine le comportement du circuit. Si $Load = 1$ sur un front montant de l'horloge alors la sortie de données prend la valeur présente à l'entrée.

Q3 Nous avons à notre disposition 8 circuits qui font office de registres 32 bits. Proposez une implémentation du fichier à 8 registres décrit dans cet exercice à l'aide de portes logiques, de circuits combinatoires classiques et bien évidemment de 8 registres 32 bits.