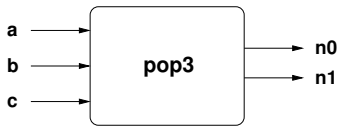
Devoir Surveillé no 2

Durée : 1h30 — Sans document

1 Circuit combinatoire simple

On souhaite construire un circuit combinatoire à trois entrées (de 1 bit chacune) et une sortie (codée sur 2 bits) réalisant la fonction « population » sur 3 bits (notée `pop3` sur le schéma), qui est le nombre d'entrées valant 1. Par exemple, si a et b sont à 1 et c est à 0, il y a 2 entrées à 1, et donc n_1 est à 1 et n_0 est à 0 (2 codé en binaire).



Q1 (échauffement) Donnez la table de vérité de la fonction `pop3`. À quel circuit connu cela vous fait-il penser ?

Q2 Proposez une implémentation du circuit `pop3` à l'aide de portes logiques simples.

Q3 Expliquez comment combiner deux circuits `pop3` et un additionneur pour obtenir un circuit `pop7`.

2 Assembleur Y86

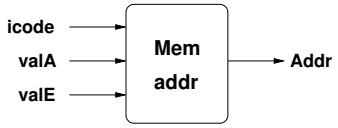
Pour les fonctions demandées ci-dessous, on utilisera le passage de paramètre standard sur la pile, mais sans utiliser `ebp`.

Q1 Écrivez une fonction `not` qui inverse chaque bit de son paramètre (on rappelle qu'Y86 ne dispose que des opérations `addl`, `subl`, `andl` et `xorl`).

Q2 Écrivez une fonction `neg`, utilisant la fonction `not`, qui transforme son paramètre x en -x *sans utiliser l'instruction* `subl`.

3 Version séquentielle du simulateur Y86

La figure ci-contre montre les entrées du circuit combinatoire « `Mem addr` » chargé de calculer l'adresse mémoire à laquelle le processeur devra charger ou écrire une valeur. Le code HCL « partiel » décrivant ce circuit est donné sous le schéma.




```
mem_addr = [
  icode in { IRMMOVL, ... } : valE;
  icode in { IPOPL, ... } : valA;
  # Others don't need an address
]
```

Q1 Expliquez le code source HCL fourni (ne vous contentez pas de paraphraser, justifiez précisément le choix de la valeur en fonction de l'*icode (instruction code)*).

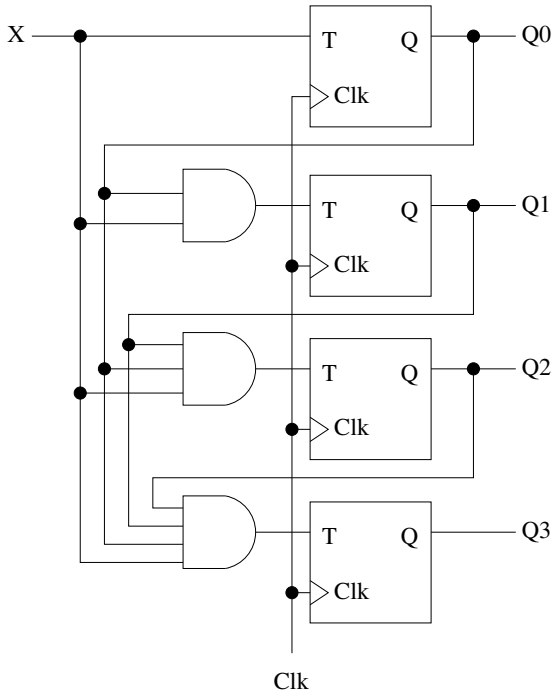
Q2 Complétez ce code source (les « ... ») pour traiter les quelques cas non gérés actuellement. Pour chaque code d'instruction ajouté, justifiez précisément le choix entre `valA` ou `valE`.

4 Version pipelinée du simulateur Y86

Q1 Les instructions CALL et RET ont-elles toutes deux besoin d'injecter des bulles dans le pipeline ? Expliquez pourquoi.

5 Circuit séquentiel

Le circuit ci-dessous utilise quatre bistables T.



Dans un bistable T, lors du front montant d'horloge la sortie Q change d'état si l'entrée T (*Toggle* = bascule) est à 1, et reste dans le même état que précédemment si T est à 0.

On suppose qu'à l'initialisation on a l'état $Q_0 = Q_1 = Q_2 = Q_3 = 0$.

Q1 On suppose pour l'instant que l'entrée X est maintenue en permanence à 1. Dressez une table des états Q_i en fonction du temps (une ligne par période d'horloge). Expliquez l'astuce utilisée dans ce circuit pour obtenir ce résultat.

Q2 Que se passe-t-il lorsque l'on maintient l'entrée X à 0 ? Comment appeler cette entrée ?

Q3 Expliquez comment fabriquer un bistable D à partir d'un bistable T.