Bordeaux INP
ENSEIRB
MATMECA

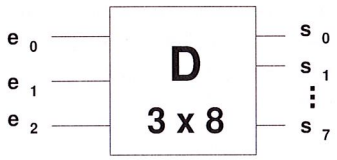
Année	2017-2018	Type	Examen Session 1
Filière	Informatique		
Code UE	MI 101	Épreuve	Microprocesseurs
Date	19/01/2018	Documents	Autorisés
Début	10h30	Durée	2h

1 Question de cours

Indiquez le principe général de fonctionnement d'un cache de données. Sur quelles hypothèses de localité est-il fondé ?

2 Circuits combinatoires

On souhaite construire un décodeur à trois entrées et huit sorties, tel qu'illustré ci-contre.

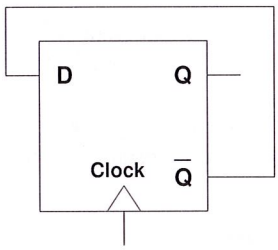


Q1 À l'aide de portes logiques élémentaires, donnez une implémentation de ce décodeur 3×8.

Q2 Donnez une implémentation de ce circuit s'appuyant sur deux décodeurs 2×4 (ainsi que quelques circuits supplémentaires).

3 Circuits séquentiels

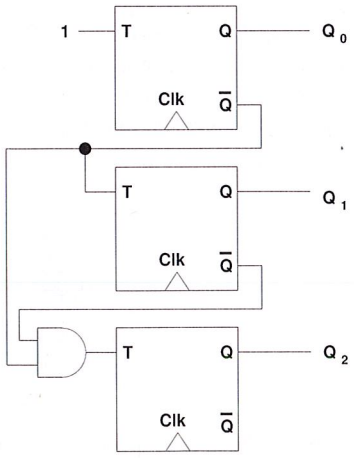
Le circuit ci-contre utilise un bistable D muni d'une sortie «inversée» $\bar{Q}$ qui vaut toujours l'inverse de Q. Ici, on a relié la sortie $\bar{Q}$ à l'entrée D. On suppose qu'initialement, la sortie Q vaut 0 (et donc $\bar{Q}$ vaut 1).



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» acceptant en entrée un signal d'horloge ainsi qu'un bit de commande (0 = normal, 1 = slow) et fournissant en sortie soit le signal d'horloge inchangé (mode *normal*) soit un signal d'horloge de période deux fois plus longue (mode *slow*).

Le circuit ci-contre est constitué de trois « bistables T ». Dans un bistable T, **lors du front montant d'horloge**, la sortie Q ne change d'état que si l'entrée T (pour *Toggle*) est à 1. Sinon, le bistable ne change pas d'état.



Q3 Proposez une implémentation d'un bistable T en utilisant un bistable D.

Q4 On s'intéresse au circuit ci-contre et on suppose qu'initialement, les trois bistables ont pour état 0 ($Q_0 = Q_1 = Q_2 = 0$). Tracez un chronogramme, sur huit périodes d'horloge, sur lequel vous indiquerez soigneusement l'évolution des valeurs de Q_0, Q_1 et Q_2. Considérant que $Q_2Q_1Q_0$ est un nombre sur trois bits, expliquez ce que fait ce circuit.

4 Programmation Y86

Donnez une version assembleur Y86 de la fonction C `xchg` ci-contre. On pourra simplifier le code en utilisant simplement un registre pour mémoriser la variable locale `tmp`.

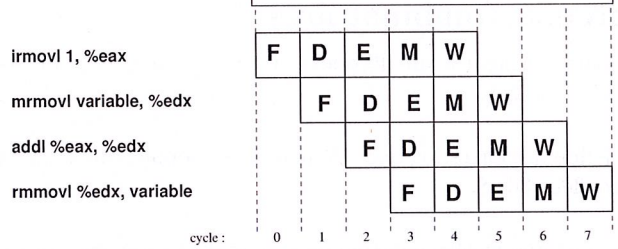
```
void xchg (int *a, int *b)
{
   int tmp = *a;
   *a = *b;
   *b = tmp;
}
```

5 Version pipelinée du simulateur Y86

On considère la version pipelinée du processeur Y86 à 5 étages (*Fetch*, *Decode*, *Execute*, *Memory* et *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 1, %eax
mrmovl variable, %edx
addl %eax, %edx
rmmovl %edx, variable
```

Q1 Si ces instructions se suivaient « idéalement » dans le pipeline, comme illustré sur le chronogramme ci-contre, l'exécution du programme durerait 8 cycles. Expliquez pourquoi une telle exécution serait incorrecte.



Q2 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 à chaque cycle.

Q3 En supposant maintenant une version du processeur capable de rétro-propager des valeurs des étages *E*, *M* ou *W* vers l'étage *D*, redessinez le chronogramme. Y a-t-il encore apparition de *bulles* ? Expliquez.

Q4 Cela change-t-il quelque chose d'inverser l'ordre des deux premières instructions dans le programme d'origine ? Aidez-vous d'un chronogramme pour répondre.

6 Boucles

On s'intéresse maintenant aux programmes contenant des boucles. L'exemple ci-contre illustre comment réaliser une boucle de 10 itérations. Pour réduire le nombre d'instructions au sein de la boucle, on peut s'inspirer des processeurs x86 et envisager d'ajouter une instruction `loop <adresse>` au processeur Y86. Cette instruction décrémente le registre `ecx`, puis réalise un saut à l'adresse `label` si la nouvelle valeur d'`ecx` est différente de zéro. Dans le programme ci-contre, on pourrait donc remplacer les instructions `isubl` et `jne` par une seule instruction : `loop boucle`.

```
        irmovl 10, %ecx
boucle:
        ...
        ...
        isubl 1, %ecx
        jne boucle
```

Q1 Proposez comment devraient être calculées les valeurs suivantes, classées ici suivant l'étage où elles sont calculées :

Étage	*Valeurs calculées*
Fetch	ValP
Decode	SrcA, SrcB
Execute	AluA, AluB, AluFun
Memory	Read, Write, Address, Data
Write Back	DstE, DstM, NextPC

Note : On supposera que le processeur « prédit » systématiquement que les branchements seront effectués.