Université Bordeaux 1, Sciences et Technologies
UFR Informatique et Mathématiques

Licence d'Informatique
Session de septembre 2000

Architecture et Microprocesseurs
Le 13 septembre 2000

Durée : trois heures.

Tous documents autorisés sauf photocopies de livres. Votre voisin n'est pas un document.

Rappel : Soyez bref et clair ! Personne n'est impressionné par la quantité !

Bon courage !

Exercice 1. (7p)

Pour une application nécessitant un calcul de minutes et de secondes, on souhaite réaliser un compteur qui compte de 0 à 5, mais dont le résultat est donné directement en code 7-segments (le type d'affichage que vous avez sur les horloges numériques, etc). De plus, on souhaite une entrée supplémentaire. Quand cette entrée a la valeur `0`, le compteur ne doit pas compter, et quand elle a la valeur `1`, il doit compter normalement. Afin de simplifier les choses, on va supposer que pour afficher chiffre `0`, il faut allumer les segments numéros `0`, `1`, `2`, `4`, `5` et `6`; pour le chiffre `1`, il faut allumer les segments numéro `2` et `6`; pour le chiffre `2`, il faut allumer les segments numéro `1`, `2`, `3`, `4` et `5`; pour le chiffre `3`, il faut allumer les segments numéro `1`, `2`, `3`, `5` et `6`; pour le chiffre `4`, il faut allumer les segments numéro `0`, `2`, `3` et `6`; et pour le chiffre `5`, il faut allumer les segments numéro `0`, `1`, `3`, `5` et `6` (voir annexe 1 pour la numérotation des segments).

a. Donner la table d'états d'un circuit séquentiel avec une entrée et sept sorties réalisant le compteur.

b. En utilisant la méthode générale (sans minimisation), réaliser le circuit combinatoire pour l'entrée du bistable correspondant au segment numéro `2`

c. En utilisant la méthode de votre choix, réaliser un circuit combinatoire pour l'entrée du bistable correspondant au segment numéro `4`.

Exercice 2. (6p)

Les langages de programmation comme le C ont souvent besoin d'accéder aux éléments d'un tableau. Si l'adresse du début du tableau est connue à la compilation, alors pour adresser l'élément numéro `i`, il faut additionner une constante (l'adresse du début) avec une variable (la valeur de la variable `i`). Pour permettre ce type d'adressage dans notre architecture d'exemple, on peut imaginer que la valeur de la variable `i` soit contenue dans le registre d'adresses, et que l'adresse du début du tableau soit donnée après le code de l'instruction. On appelle ce type d'adressage *indexé*.

Pour que cet adressage soit intéressant, il faut au moins une instruction `load`, par exemple `LDIX`, et une instruction `store`, par exemple `STIX`, utilisant l'adressage indexé.

a. Modifier l'architecture donnée en annexe 2 pour que les deux nouvelles instructions soient réalisables.

b. Réaliser le micro-programme de l'instruction `LDIX`

Exercice 3. (7p)

Certaines applications de type temps-réel nécessitent un temps de réponse assez rapide. Sur une machine multi-programmée (avec plusieurs processus en exécution simultanément), on a donc intérêt à donner à chaque processus peu de temps d'exécution avant d'affecter le processeur au processus suivant. Par contre, en ce qui concerne la mémoire cache, on a plutôt intérêt à laisser un processus tourner le plus de temps possible.

Si l'on suppose que :

- chaque processus exécute un programme dont le code est une boucle infinie de la taille du cache (ou un peu moins) qui utilise très peu de données,
- la taille du cache est 64ko,
- chaque instruction est codée sur 4 octets,
- une instruction nécessite 2ns à son exécution si elle est dans le cache, et 20ns sinon (une ns $= 10^{-9}$s)
- à chaque changement de processus, le cache est vidé.

Quelle est la durée minimale d'exécution d'un processus afin que le temps pour charger le cache représente au plus 1% du temps d'exécution. La réponse n'a pas besoin d'être exacte, mais elle doit être justifiée par un raisonnement correct.