Université Bordeaux 1, Sciences et Technologies
UFR Mathématiques et Informatique

Licence d'Informatique
Session de janvier 2000

# Architecture et Microprocesseurs

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. (5p)**

Le code de Gray est conçu pour qu'un compteur pour ce code ne change qu'un seul bit à chaque étape. Un tel compteur de 3 bits compte alors comme ceci : `000`, `001`, `011`, `010`, `110`, `111`, `101`, `100`, `000`, `...`, alors qu'un compteur normal compte de cette façon : `000`, `001`, `010`, `011`, `100`, `101`, `110`, `111`, `000`, `...`.

a. Donner la table d'états d'un circuit séquentiel avec une entrée et trois sorties, tel que si l'entrée est égale à 1, alors le circuit se comporte comme un compteur normal à 3 bits, et si l'entrée est égale à 0, alors le circuit se comporte comme un compteur Gray à 3 bits.

b. En utilisant la méthode générale (sans minimisation), réaliser le circuit combinatoire pour l'entrée du bistable correspondant au bit le moins significatif.

**Exercice 2. (5p)**

Les systèmes d'exploitation modernes utilisent des bibliothèques partagées. Le code (les instructions) d'une telle bibliothèque n'est pas chargé à une adresse fixe, mais une adresse potentiellement différente dans chaque application utilisant la bibliothèque.

Pour que le code marche indépendamment de sa position (on parle de PIC : position-independent code), on rajoute une nouvelle famille d'instructions de sauts. L'argument de ces instructions n'est pas interprété comme une adresse absolue, mais comme une *adresse relative* à additionner au compteur ordinal (PC) pour obtenir l'adresse absolue.

a. Modifier l'architecture donnée en annexe pour que la nouvelle famille d'instructions soit réalisable.

b. Réaliser le micro-programme de l'instruction JRAL (Jump Relative ALways), c'est à dire le saut non conditionnel relatif.

**Exercice 3. (5p)**

Le protocole d'appel de sous-programmes discuté en cours a l'inconvénient que l'appelé ignore le nombre d'arguments fournis par l'appelant (c'est pour ça que c'est l'appelant qui est responsable de dépiler les arguments). Proposer un protocole d'appel avec deux registres (SP (stack pointer) et BP (base pointer)) tel que, quand l'appelé commence son exécution, la différence entre BP et SP correspond au nombre d'arguments passés.

Pour que le protocole soit complet, il faut déterminer :
- à quel moment l'appelant empile l'adresse de retour,
- à quel moment l'appelant calcule et empile les arguments,
- comment l'appelé alloue des variables locales,
- comment l'appelé trouve les arguments et les variables locales,
- qui dépile les arguments (l'appelant ou l'appelé),
- comment l'appelé retourne à l'appelant.

Attention : il n'y a pas de circuit à réaliser, et vous avez le droit d'inventer des instructions si nécessaire. Dessiner la pile avec les arguments, les variables locales, l'adresse de retour, et la valeur de SP et BP à chaque instant du protocole.

**Exercice 4. (5p)**

Réaliser un bistable D en n'utilisant que des portes *non-et* et des inverseurs. Indication : remplacer les portes *non-ou* des deux bascules SR par des portes *et* et des inverseurs, puis simplifier autant que possible.