Université Bordeaux 1, Sciences et technologies
UFR Informatique et Mathématiques

Licence d'informatique
Session de septembre 2003

Architecture et Microprocesseurs (AMP)
Le 13 septembre 2003

Durée : trois heures

Tous documents autorisés sauf photocopies illégales 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 (7 p)

Une rue principale est traversée par une piste cyclable peu fréquentée. Pour permettre aux cyclistes de traverser la rue, on souhaite installer un feu de circulation. À cause de la faible fréquentation de la piste cyclable et la circulation assez dense de la rue, le feu doit être vert pour la rue et rouge pour la piste presque tout le temps.

Les cyclistes disposent d'un bouton pour pouvoir traverser. Lorsqu'un cycliste appuie sur le bouton (même momentanément), un cycle d'événements se déclenche pour temporairement donner le feu vert au cycliste. On peut décrire ce cycle par une suite de couples de valeurs. Le premier élément du couple indique la couleur du feu pour la rue et le deuxième la couleur du feu pour la piste. L'état normal (vert pour la rue, rouge pour la piste) est donc indiqué par le couple (VR) (V=vert, R=rouge, J=jaune). La suite d'états déclenchée par le bouton sera donc :

(JR), (RR), (RV), (RV), (RJ), (RR), (VR)

La transition d'un état à l'état suivant se fera au rythme de l'horloge que pour notre exemple sera d'environ 3 secondes par période. On remarque la double occurrence de l'état (RV) dans la suite pur permettre au cycliste un peu plus de temps pour traverser.

Le fabricant de feux de circulation propose une configuration standard pour piloter un carrefour normal. Cette configuration est livré avec trois signaux d'entrée (x, y, z) pour déterminer l'état des feux. Voici la signification de ces trois entrées :

x	y	z	signification
0	0	0	tout les ampoules éteintes
0	0	1	deux feu rouge
0	1	0	jaune pour la rue, rouge pour la piste
0	1	1	jaune pour la piste, rouge pour la rue
1	0	0	vert pour la rue, rouge pour la piste
1	0	1	vert pour la piste, rouge pour la rue
1	1	0	jaune clignotant pour la rue et pour la piste
1	1	1	tout les ampoules allumées (pour les testes)

L'exercice consiste à créer le circuit séquentiel permettant le pilotage du feux. Pour cela, mise à part les trois signaux (x, y, z) pour piloter le feu proprement dit, vous disposez de l'entrée correspondant au bouton (b) disponible aux cyclistes. Vous avez également besoin d'une sortie "cachée" (appelée c) qui aura la valeur '0' pendant le premier entre les deux états (RV) et '1' pendant le deuxième.

a. (4p) Remplir la table d'états donnée en annexe.

Veuillez effectuer la modification directement sur la feuille donnée en annexe. Ne pas oublier de fournir cette feuille (munie de votre numéro d'anonymat) avec votre copie.

(b) [**3p**] Réaliser le circuit correspondant au bistable x.

Exercice 2 (13 p)

Certains langages de programmation manipulent des *données typées*, à savoir qu'il est possible de déterminer en regardant une donnée s'il s'agit d'un nombre, d'un caractère, d'un pointeur ou d'autres choses. On appelle un tel langage *dynamiquement typé*.

Pour aider le concepteur du compilateur d'un tel langage, on souhaite lui proposer un ensemble d'instructions supplémentaires permettant des opérations arithmétiques entre deux opérandes ainsi typés.

Pour cela, on représente un petit entier i par la représentation habituelle de $2i$. L'entier 13 sera donc représenté par 00011010, à savoir la représentation habituelle de 26. De cette manière, chaque petit entier a toujours le bit le moins significatif égal à '0'. Si le bit le moins significatif n'est pas égal à '0', il s'agit d'un autre type de donnée, et toute tentative d'effectuer une opération arithmétique la dessus doit déclencher une erreur.

On remarque que les opération arithmétiques habituelles (addition et soustraction) marche également sur ce type de données typées. Aucune modification de l'unité arithmétique est logique ne sera donc nécessaire.

(a) [**4p**] Modifier l'architecture donnée en annexe pour permettre une telle famille d'instructions. Ces instructions diffèrent des opérations arithmétiques habituelles par le fait qu'elles acceptent un *argument* correspondant à une *adresse* dans le programme. Si les arguments sont du bon type, l'instruction effectue l'opération sans se servir de l'adresse. Si par contre une erreur de typage est détectée, l'instruction fonctionne comme une instruction de saut conditionnelle dans le sens qu'elle provoque un saut à l'adresse donnée en argument (l'opération arithmétique est éventuellement effectuée quand même).

Veuillez effectuer la modification directement sur la feuille donnée en annexe. Ne pas oublier de fournir cette feuille (munie de votre numéro d'anonymat) avec votre copie.

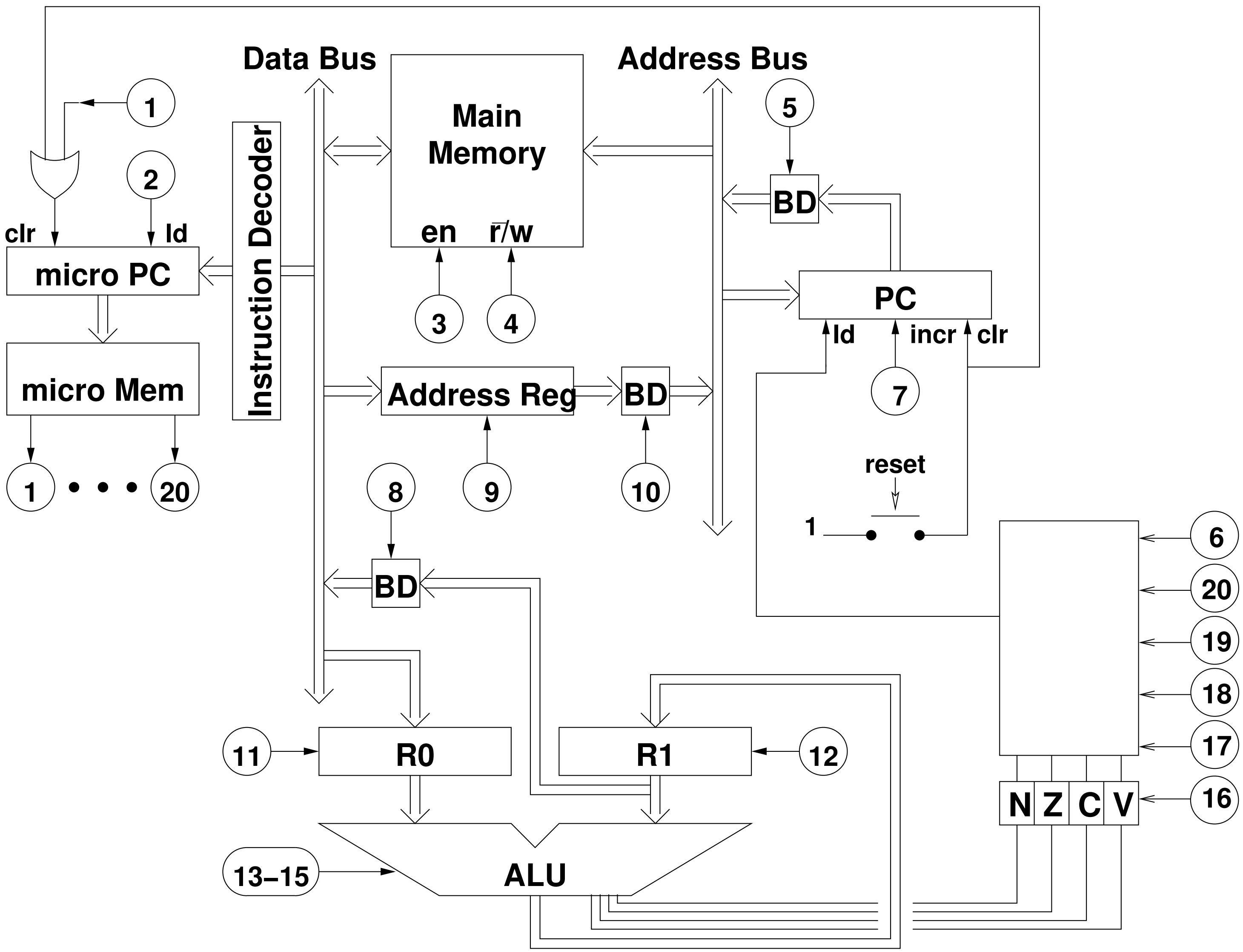
(b) [3p]

Donner le micro programme des deux instructions $ADDT$ (addition typée) et $SUBT$ (soustraction typée).

(c) [6p]

En supposant qu'une *fonction* `error` existe déjà (et qui doit donc être appelée comme une fonction normale), donner le code généré par un compilateur pour un langage dynamiquement typé à partir d'une expression comme `z = x + y`.

b	x	y	z	c	x'	y'	z'	c'
0	0	0	0	0				
0	0	0	0	1				
0	0	0	1	0				
0	0	0	1	1				
0	0	1	0	0				
0	0	1	0	1				
0	0	1	1	0				
0	0	1	1	1				
0	1	0	0	0				
0	1	0	0	1				
0	1	0	1	0				
0	1	0	1	1				
0	1	1	0	0				
0	1	1	0	1				
0	1	1	1	0				
0	1	1	1	1				
1	0	0	0	0				
1	0	0	0	1				
1	0	0	1	0				
1	0	0	1	1				
1	0	1	0	0				
1	0	1	0	1				
1	0	1	1	0				
1	0	1	1	1				
1	1	0	0	0				
1	1	0	0	1				
1	1	0	1	0				
1	1	0	1	1				
1	1	1	0	0				
1	1	1	0	1				
1	1	1	1	0				
1	1	1	1	1				


Data Bus
Main Memory
Address Bus
1
2
clr
ld
micro PC
micro Mem
1
20
Instruction Decoder
en
r/w
3
4
5
BD
PC
ld
incr
clr
7
Address Reg
BD
8
9
10
reset
1
BD
6
20
19
18
17
16
N
Z
C
V
11
R0
R1
12
13–15
ALU