



Module : Architecture des Ordinateurs

Responsable cours : Mokrani Hocine

Filière : Licence Informatique

Documents : Non autorisés

Rattrapage 2018-2019
(01 h 00 min)

Question de cours : (11 points)

1. Quelle différence il-y-a-t-il entre un accès séquentiel et un accès direct à une mémoire?	(1 pt)
<p>Accès séquentiel : Le plus lent, Pour accéder à une information particulière, on est obligé de parcourir toutes celle qui la précèdent. (0.5 pts)</p> <p>Accès Direct : Le plus rapide, Les données ont une adresse propre, on y accède directement. (0.5 pts)</p>	
2. Donner la différence entre mémoire physique et mémoire logique.	(1 pt)
<p>La mémoire logique est la façon dont le processeur (ou le programmeur) voit la mémoire (physique). (1 pt)</p> <p>Par exemple, sur le processeur MIPS, la mémoire est définie comme un ensemble de 8 octets consécutifs dont la première adresse est 0 et la dernière adresse est $2^{(taille\ bus\ d'adresse)} - 1$. Physiquement, cette mémoire peut être organisée d'une manière unidimensionnelle, bidimensionnelle, ou avec plusieurs boîtiers mémoire.</p>	
3. Citez quatre modes d'accès à la mémoire centrale pour le processeur MIPS R3000 ?	(2 pts)
<p>Adressage immédiat : (0,25 pt)</p> <p>Opérandes est la valeur utilisée par l'instruction (Exemple d'instruction : addi \$1,\$2,16). (0,25 pt)</p> <p>Adressage par registre : (0,25 pt)</p> <p>Opérandes est l'adresse de registre contenant la valeur (Exemple add \$1,\$2,\$3). (0,25 pt)</p> <p>Adressage par base : (0,25 pt)</p> <p>Opérandes est un décalage par rapport à une adresse dans l'un des registres. La somme de l'opérande et la valeur de registre permet de trouver l'adresse contenant la valeur. (Exemple : lw \$1, 100(\$2)) (0,25 pt)</p> <p>Adressage relatif selon \$PC : (0,25 pt)</p> <p>Opérandes est un décalage par rapport à une adresse dans le registre \$PC. La somme de l'opérande et la valeur de registre permet de trouver l'adresse contenant la prochaine instruction à exécuter. (Exemple : bne \$1, \$2, 100). (0,25 pt)</p>	
4. Sur le processeur MIPS R3000 , combien de bits il faut réserver sur le registre d'adresse, pour découper une mémoire de 4Mo (adressable par octet) en plusieurs segments de 4Ko ?	(2pts)
<p>$nb_segment = taille_memoire / taille_segment$ (0.5 p)</p> <p>$nb_segment = 4 * 2^{20} / 4 * 2^{10} = 2^{10}$ (0.5 p)</p> <p>$nb_bit = \log(2^{10}) = 10\ bits$ (0.75 p)</p> <p>Donc, Il faut réserver les 10 bits de poids fort du registre d'adresse. (0.25 p)</p>	
5. Quel est la différence entre une instruction assembleur et une pseudo-instruction assembleur ?	(1 pt)
<p>Une instruction assembleur : est un mot mémoire qui comporte une commande avec ses paramètres compréhensible par le processeur. (0.5 pts)</p> <p>Une pseudo-instruction : est une instruction qui comporte une commande paramétrée qui</p>	

remplace une ou plusieurs instructions assembleur. Le but de création des pseudo-instructions est de simplifier l'écriture et la compréhension du code par l'être humain.(0.5 pts)												
6. Citer quatre registres différents du processeur MIPS et décrire leur rôle en une courte phrase ?		(2tps)										
<p>((0.25 pts x 4) pour chaque registre (0.25 pts x 4) pour chaque explication).</p> <p>\$ra : registre d'adresse, contient l'adresse de retour.</p> <p>\$sp : registre de pile, contient l'adresse du sommet de la pile.</p> <p>\$fp, \$PC, \$RI, \$ZERO</p>												
7. Citer les différences principales entre une architecture CISC et une architecture RISC ?		(2pts)										
<table border="1"> <thead> <tr> <th>CISC (1 point)</th> <th>RISC (1 point)</th> </tr> </thead> <tbody> <tr> <td>1. Instructions complexes. (plusieurs cycles par instruction).</td> <td>1. Instructions simple (Un cycle par instruction généralement).</td> </tr> <tr> <td>2. Instuctions de taille variables.</td> <td>2. Instruction de taille fixe (1 mot)</td> </tr> <tr> <td>3. Instructions séquentielles généralement.</td> <td>3. Pipeline avancé.</td> </tr> </tbody> </table> <p>On compte aussi les avantages et inconvénients comme différences l'idée est que les étudiants doivent trouver 3 différences pour chaque type d'architecture. Voici quelques avantages et inconvénients :</p> <table border="1"> <tbody> <tr> <td> <ul style="list-style-type: none"> ⊕ Programmation de plus haut niveau. ⊕ Compilation moins complexe ⊕ Programmation plus compacte. moins d'occupation en Mémoire et à l'exécution. ⊖ Complexité du processeur. ⊖ Fréquence d'horloge réduite. ⊖ Instructions lentes </td> <td> <ul style="list-style-type: none"> ⊕ Mode d'adressage réduit. ⊕ Pipeline plus simple et plus efficace. Traitement efficace. ⊕ Fréquence d'horloge réduite. ⊕ Instruction rapides. ⊖ Beaucoup de registre ⊖ Programme plus volumineux ⊖ Compilation plus complexe. </td> </tr> </tbody> </table>			CISC (1 point)	RISC (1 point)	1. Instructions complexes. (plusieurs cycles par instruction).	1. Instructions simple (Un cycle par instruction généralement).	2. Instuctions de taille variables.	2. Instruction de taille fixe (1 mot)	3. Instructions séquentielles généralement.	3. Pipeline avancé.	<ul style="list-style-type: none"> ⊕ Programmation de plus haut niveau. ⊕ Compilation moins complexe ⊕ Programmation plus compacte. moins d'occupation en Mémoire et à l'exécution. ⊖ Complexité du processeur. ⊖ Fréquence d'horloge réduite. ⊖ Instructions lentes 	<ul style="list-style-type: none"> ⊕ Mode d'adressage réduit. ⊕ Pipeline plus simple et plus efficace. Traitement efficace. ⊕ Fréquence d'horloge réduite. ⊕ Instruction rapides. ⊖ Beaucoup de registre ⊖ Programme plus volumineux ⊖ Compilation plus complexe.
CISC (1 point)	RISC (1 point)											
1. Instructions complexes. (plusieurs cycles par instruction).	1. Instructions simple (Un cycle par instruction généralement).											
2. Instuctions de taille variables.	2. Instruction de taille fixe (1 mot)											
3. Instructions séquentielles généralement.	3. Pipeline avancé.											
<ul style="list-style-type: none"> ⊕ Programmation de plus haut niveau. ⊕ Compilation moins complexe ⊕ Programmation plus compacte. moins d'occupation en Mémoire et à l'exécution. ⊖ Complexité du processeur. ⊖ Fréquence d'horloge réduite. ⊖ Instructions lentes 	<ul style="list-style-type: none"> ⊕ Mode d'adressage réduit. ⊕ Pipeline plus simple et plus efficace. Traitement efficace. ⊕ Fréquence d'horloge réduite. ⊕ Instruction rapides. ⊖ Beaucoup de registre ⊖ Programme plus volumineux ⊖ Compilation plus complexe. 											

Exercice 1: (6 points)

Ecrire en assembleur **MIPS** le code de la fonction **prog** suivante :

```
int prog(int nb)
{
    if(nb > 1)
    {
        return (nb * prog(nb-1));
    }
    else
    {
        return (1);
    }
}
```

Remarque

Le paramètre « **nb** » est alloué au registre « **\$4** ». La valeur de retour de la fonction doit être dans le registre « **\$2** ». Utilisez la pile pour sauvegarder les valeurs des registres, pour éviter l'écrasement des registres lors de la récursivité.

(1 pt)	{	prog:	
		li \$1,1	
		bgt \$4, \$1, recursive	#si nb>0, appel récursif, sinon retourne 1
		li \$2, 1	# prog(0)= 1
(1 pt)	{	j \$ra	# retourne (adresse de retour dans \$ra)
		recursive:	
		sub \$29, \$29, 8	#sauvegarder l'adresse de retour+paramètre
		sw \$31, 0 (\$sp)	#sauvegarder adresse de retour
(1 pt)	{	sw \$4, 4 (\$sp)	# sauve le paramètre nb
		sub \$4, \$4, 1	# nb-1
		jal prog	# appel récursif, le résultat est dans \$2
(1 pt)	{	lw \$4, 4(\$sp)	# restaure le paramètre
		lw \$31, 0 (\$sp)	# restaure adresse de retour
(1 pt)	{	add \$29, \$29, 8	# libérer la place de la pile.
		mul \$2, \$4, \$2	# multiplier par le paramètre
		j \$ra	# retourne (adresse de retour dans \$ra)
		(1 pt pour la propreté et organisation du code)	((- 0.5 pt) pour l'absence de commentaire)

Exercice 3 : (3 points)
 Soit le circuit séquentiel de la figure 1.

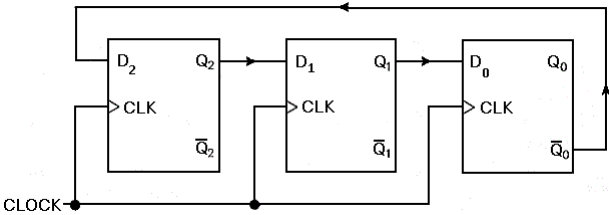


Figure 1 : Schéma du circuit

Horloge	D	Q(t+1)
	0	0
	1	1

Figure 2 : Table d'états d'une bascule D.

1. Quel est le comportement du circuit de la figure (**Figure 1**) en supposant qu'initialement $Q_0 = 0$, $Q_1 = 0$ et $Q_2 = 0$, c'est-à-dire, quelles sont les séquences des états Q_0 , Q_1 , Q_2 suivantes? Justifiez votre réponse.

(1 pt)
- Suggestion :** faites un tableau avec comme en-tête $Q_2(t)$, $Q_1(t)$, $Q_0(t)$, D_2 , D_1 , D_0 , $Q_2(t+1)$, $Q_1(t+1)$, $Q_0(t+1)$.
2. Donner l'automate d'états de ce circuit ?

(1 pt)
- Note :** Les sorties primaires et secondaires du circuit sont les valeurs de sortie des bascules Q_0 , Q_1 et Q_2 .
3. Que fait donc ce circuit ? Expliquez en une courte phrase.

(1 pt)

1. Table d'état.

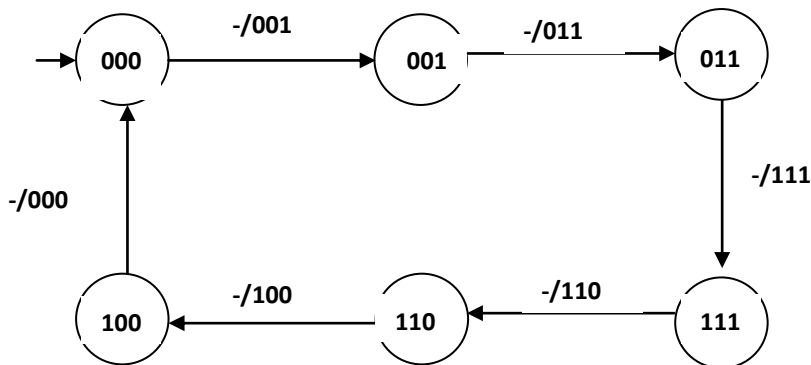
(1 pt)

$Q_2(t)$	$Q_1(t)$	$Q_0(t)$	D_2	D_1	D_0	$Q_2(t+1)$	$Q_1(t+1)$	$Q_0(t+1)$
0	0	0	1	0	0	1	0	0
1	0	0	1	1	0	1	1	0
1	1	0	1	1	1	1	1	1
1	1	1	0	1	1	0	1	1
0	1	1	0	0	1	0	0	1
0	0	1	0	0	0	0	0	0

2. Automate d'état.

(1,5 pt)

- Les états sont codés par : Q_2 le bit de poids faible, Q_0 le bit de poids fort.
- Les transitions sont codées par : $-/ Q_0 Q_1 Q_2$. Il n'existe pas une entrée primaire.
(0,75 pt pour les états, 0,75 pour les transitions).



3. Le circuit fait le remplissage des cases de droit à gauche par des uns suivies par le remplissage des cases de droit à gauche par des zéros, d'une manière infinie.

(0,5 pt)

Bon courage