EXERCICES D'ARCHITECTURE DES ORDINATEURS

CHAPITRES 1&2

1. CONVERSION DANS D'AUTRES BASES

Écrire 10110110_2 en décimal.

Écrire 3456 en binaire, puis en hexadécimal.

Convertir 1011 1100 0000 1000 1100 en décimal

2. CODAGE EN COMPLEMENT A 2

Sur 8 bits, donner les codages en complément à 2, lorsque c'est possible, des nombres suivants :

0, -1, +10, -10, +64, -64, +127, -127, +200

3. ARITHMETIQUE BINAIRE

Sur 8 bits, effectuer la somme en binaire : $1011\ 0101_2 + 1100\ 0011_2$ et interpréter le résultat en écriture décimale, en arithmétique signée et non signée. Indiquer dans chaque cas s'il y a débordement.

Comparer 10101100 et 11001100 en signé et non signé

Même question pour la somme : $111\ 0101_2 + 1100\ 0011_2$

1. ALGEBRE DE BOOLE

Simplifier l'expression suivante, en utilisant les théorèmes de l'algèbre de Boole :

$$A.\overline{B}.C + A.\overline{C}.D + A.C + \overline{D}$$

4. ALGEBRE DE BOOLE : QUINE MC CLUSKEY

Simplifier l'expression suivante, en utilisant l'algorithme de Quine McCluskey :

$$\sum(1, 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31)$$

5. ALGEBRE DE BOOLE : GLITCH

Enumérer tous les glitches que peut présenter l'expression suivante :

$$S = A\ \overline{B}\ D + B\ C\ D + A\ C\ \overline{D}$$

Proposez une expression modifiée qui supprime tous ces glitches.

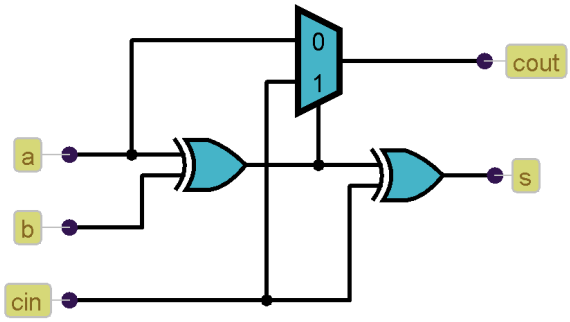
6. ALGEBRE DE BOOLE

+ Ecrire les équations associées à un schéma

+ opposé d'une somme de mintermes -> sommes de tous les autres

- équation >= 9, < 12

Prouver algébriquement (par un calcul) que le schéma suivant est un additionneur complet :



7. ALGEBRE DE BOOLE

Réécrire l'expression suivante, uniquement avec l'opérateur complet NAND :

XOR(A,B)

8. CIRCUIT COMBINATOIRE : SYNTHESE PAR TABLE DE VERITE

On souhaite concevoir un circuit combinatoire qui calcule la valeur absolue en binaire pur sur 4 bits x,y,z,t d'un nombre codé en complément à 2 sur 4 bits, a,b,c,d. Cas particulier : si abcd=1000 (-8), alors xyzt=0000 (car +8 n'est pas codable sur 4 bits)

Donner la table de vérité de ce circuit combinatoire, puis fournir les équations de x,y,z,t en fonction de a,b,c,d en les simplifiant au besoin à l'aide de tables de Kanaugh.

9. CIRCUIT COMBINATOIRE : SYNTHESE FONCTIONNELLE

On souhaite concevoir un circuit combinatoire qui commande l'ouverture d'une porte de jardin publique, ouverte en hiver de 10h à 18h et en été de 8h à 20h. L'heure est codée sur 5 bits, sous forme d'un nombre entier H compris entre 0 et 23 ; la saison est codée sous forme d'un signal S (0=hiver, 1=été).

Concevoir un circuit d'interface `PORTE(H[4..0], S : M)` tel que M=1 si et seulement si :

- S = 0 et 10 ≤ H ≤ 18, ou
- S = 1 et 8 ≤ H ≤ 20

On concevra le circuit de façon fonctionnelle, en combinant uniquement :

- 2 comparateurs non signés sur 5 bits
- 2 multiplexeurs 2 :1 opérant sur des bus de 5 bits
- une porte ET à deux entrées

10. CIRCUIT COMBINATOIRE : CONCEPTION MODULAIRE

Construire un encodeur de priorité à 8 entrées avec 2 encodeurs de priorité à 4 entrées. Les circuits ont l'interface :

```
prioencoder4(e[3..0] : num[1..0], act)
```

```
prioencoder8(e[7..0] : num[2..0], act)
```

11. CIRCUIT COMBINATOIRE : CONCEPTION MODULAIRE

Construire un multiplexeur 8 :1 avec 2 multiplexeurs 4 :1 et un multiplexeur 2 :1

12. CIRCUIT COMBINATOIRE : CONCEPTION MODULAIRE

Construire un décodeur 3 : 8 avec 2 décodeurs 2 : 4

On utilisera des décodeurs avec une entrée EN (enable)

13. CIRCUIT COMBINATOIRE : CONCEPTION MODULAIRE

Construire un comparateur non signé de 8 bits, d'interface :

```
ucmp8(a[7..0], b[7..0] : sup, eq)
```

avec 2 comparateurs non signés de 4 bits, d'interface :

```
ucmp4(a[3..0], b[3..0] : sup, eq)
```

14. MEMOIRES

Associer 4 modules mémoires de 4K mots de 4 bits, d'interface `mem4Kx4(addr[11..0], W, CE, OE : data[3..0])` pour former un module de 16K mots de 4 bits.

CHAPITRES 3&4

15. CIRCUIT SEQUENTIEL : CONCEPTION AVEC GRAPHE D'ETATS

Concevoir un circuit séquentiel synchrone d'interface `croissant(rst, clk, a[1..0] : up)` tel que up=1 si et seulement si la valeur de a[1..0] au front d'horloge est plus grande ou égale (comparaison non signée) à la valeur de a[1..0] au front d'horloge précédent (= si a[1..0] croit ou reste stable).

- dessiner le graphe de MEALY de ce circuit
- synthétiser le circuit avec des bascules D

16. CIRCUIT SEQUENTIEL : CONCEPTION FONCTIONNELLE

Concevoir de façon fonctionnelle un circuit analogue au précédent opérant sur des nombres de 4 bits avec un registre 4 bits et un comparateur non signé sur 4 bits.

17. CIRCUIT SEQUENTIEL : CONCEPTION AVEC GRAPHE D'ETATS

Concevoir un circuit de commande de chaudière d'interface :

```
module chaudiere(rst, clk, chaud, froid : m)
```

froid = 1 ssi il fait moins de 18° ; chaud = 1 ssi il fait plus de 22°

La chaudière doit se mettre en marche lorsqu'il fait froid, et doit s'arrêter lorsqu'il fait chaud

18. CIRCUIT SEQUENTIEL : ANALYSE

On considère le module SHDL suivant :

```
module reverse(rst, h, f, c : s)
    x := /tx*x + tx*/x;
    x.clk = h;
    x.rst = rst;
    tx = c*x + f*c*/y + /c*/x*y;

    y := dy ;
    y.clk = h;
    y.rst = rst;
    dy = f*/y + /c*y + /f*/c*x;

    s = y ;
end module
```

- Dessiner le schéma du circuit
- Est-ce un circuit de Moore ou de Mealy ?
- Dessiner la table de transitions
- Simplifier la table ; dessiner le graphe simplifié

On ne cherchera pas à synthétiser le circuit simplifié

19. CIRCUIT SEQUENTIEL : GRAPHE D'ETATS

On veut réaliser une serrure à code simplifiée. Le clavier est composé de 3 touches : '#', 'A', 'B' et une led s'allume lorsque la bonne séquence a été tapée au clavier. La séquence d'ouverture est : '#', 'B', 'A'. Le circuit à réaliser aura l'interface :

```
module serrure(rst, clk, dieze, a, b : ouvert)
```

On fait les hypothèse suivantes :

- l'horloge clk est suffisamment rapide pour ne manquer aucun événement sur les touches

- le clavier est conçu pour qu'à chaque instant, il y ait zéro ou une touche appuyée, jamais plusieurs

- entre 2 appuis successifs, les trois touches sont relâchées

La séquence précise d'événements sur les touches qui déclenche l'ouverture est : appui sur #, aucun appui, appui sur B, aucun appui, appui sur A. Tout appui ensuite arrête l'ouverture et commence une nouvelle séquence.

- expliciter les vecteurs des entrées et des sorties et dessiner un graphe de Moore de ce système

- simplifier éventuellement ; quelle est la taille du vecteur d'états ?

- NE PAS synthétiser ce circuit

20. CIRCUIT SEQUENTIEL : SYNTHESE

On considère la table de transitions d'un circuit synchrone pur de type Mealy, d'entrées A, B et de sortie S :

A	**B**	**état**	**état**	**S**
0	0	a	a	0
0	1	a	c	1
1	0	a	b	1
1	1	a	d	0
0	0	b	b	1
0	1	b	d	0
1	0	b	a	0
1	1	b	c	1
0	0	c	a	1
0	1	c	c	0
1	0	c	b	0
1	1	c	d	1
0	0	d	b	0
0	1	d	d	1
1	0	d	a	1
1	1	d	c	0

- donner la taille des vecteurs d'entrées, d'état, de sorties

- synthétiser ce circuit avec les bascules de votre choix, et dessiner le circuit correspondant

21. CIRCUIT SEQUENTIEL

Dessiner le graphe de MOORE d'un circuit séquentiel qui détecte la séquence 1,1 ; concevoir ce circuit à l'aide d'un registre à décalage de 2 bits.

Transformer le graphe de MOORE en graphe de MEALY, puis concevoir le circuit correspondant.

En s'inspirant du résultat obtenu, produire un circuit de MEALY qui détecte la séquence 1,0,1.

22. MODIFICATION D'UN COMPTEUR

Concevoir et donner le code SHDL d'un compteur-décompteur sur 4 bits, qui compte selon la séquence 0, 1, …, 14, 15, 14,…, 2, 1, 0, 1,…

On produira aussi en sortie un signal `down` qui indique si le compteur est en phase descendante (=1) ou ascendante (=0).

23. COMPTEUR AVEC INITIALISATION

Concevoir un compteur 4 bits synchrone d'interface `cpt4ini(rst, clk, load, ini[3..0] : s[3..0])` tel que, lorsque load = 0 il compte normalement, et lorsque load=1 la valeur du compteur est initialisée avec ini[3..0] au front d'horloge.

CHAPITRES 5

24. AJOUT D'UNE NOUVELLE INSTRUCTION

On souhaite ajouter à CRAPS une nouvelle instruction (et NON une instruction synthétique), d'écriture assembleur :

```
swap %r<i>, %r<j>
```

Cette instruction doit permuter le contenu de %r<i> et %r<j>, sans modifier aucun autre registre (utilisateur).

Proposer un nouveau code machine pour cette instruction et décrire toutes les modifications à effectuer pour l'implémenter.

25. AJOUT D'UNE NOUVELLE INSTRUCTION

On souhaite ajouter à CRAPS une nouvelle instruction (et NON une instruction synthétique), d'écriture assembleur :

```
max %rs1, %rs2, %rd
```

Cette instruction copie dans %rd la plus grande des deux valeurs des registres %rs1 ou %rs2, considérées comme signées.

Proposer un nouveau code machine pour cette instruction et décrire toutes les modifications à effectuer pour l'implémenter.

26. SEQUENCE DE MICROCOMMANDES ASSOCIEES A L'INSTRUCTION

Compléter le tableau suivant avec la suite des microcommandes à envoyer pour exécuter l'instruction :

```
ld    [%rs1+simm13], %rd
```

areg	breg	dreg	cmd_ual	oe_num	write	commentaire

27. ACCELERATION DE L'EXECUTION D'UNE INSTRUCTION

On souhaite accélérer les instructions de lecture en mémoire lorsqu'elles ont la forme particulière suivante (registre rs2 nul):

```
ld [%ri+%r0],%rj        (1)
```

%ri et %rj sont deux registres utilisateurs ; elle est habituellement écrite `ld [%ri],%rj`

- dessiner le code machine de cette instruction (les champs i et j seront indiqués de façon générique)
- dessiner le sous-graphe de séquencement des instructions ld avec 2ème opérande registre, tel qu'il a été vu en TDTP
- **Modifier ce graphe** pour accélérer les instructions du type (1).

28. AJOUT D'UN BOITIER MEMOIRE

On souhaite rajouter un boitier mémoire à la machine CRAPS, de taille 1024 mots de 32 bits, à partir de l'adresse 0x50000000 dans l'espace d'adressage du processeur. Décrire toutes les modifications à effectuer.

29. DECODAGE D'ADRESSE

On souhaite rajouter à la micromachine 8 autres leds, mappées à l'adresse 0x80000000 dans l'espace d'adressage du processeur. Décrire toutes les modifications à effectuer.

30. PROGRAMMATION DE CRAPS

On suppose que %r2=0x8000

Pour chacun des cas suivants, écrire une (seule) instruction qui effectue l'opération demandée :

- Ecrit en mémoire %r3 à l'adresse 0x7FFC
- Fait en sorte que le flag Z indique la nullité de %r1 (d'autres flags peuvent être modifiés)

31. ASSEMBLAGE

Donnez le code machine des instructions suivantes, en binaire et en hexadécimal :

```
umulcc      %r1, -3, %r2
```

```
            ld          [%r3+%r4], %r5
loop:       ba          loop
```

32. DESASSEMBLAGE

Donnez l'écriture assembleur des instructions de codes suivants :

0xC4204000

0x81C7E004

33. ACCELERATION DE L'EXECUTION D'UNE INSTRUCTION

On désire accélérer l'exécution des instructions ld avec 2ème opérande immédiat.

Idée : faire faire simultanément par l'UAL le calcul de l'extension de signe de la constante ET la somme des deux opérandes pour le calcul de l'adresse mémoire.

Proposer une modification du graphe de séquencement de cette instruction ET l'ajout d'une nouvelle fonction de l'UAL (que nécessite cette méthode). On spécifiera la nouvelle fonction de l'UAL (par une formule ou une phrase) sans décrire son implémentation, et on dessinera le sous-graphe de séquencement modifié.

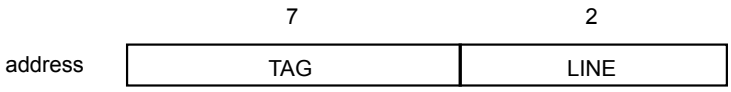
CHAPITRE 6

34. GESTION D'UN CACHE

Voici un cache de type « directed mapped » pour une mémoire de 512 mots de 16 bits, avec son contenu.

	valid	dirty	Tag (7 bits)	Data (1 word)
00	0	0	0101100	0000 0000 0000 0000
01	1	1	1110000	1000 0000 1000 0000
10	1	0	0101010	0101 0000 0000 0000
11	0	0	0000000	0000 0000 0000 1100

4 cache lines



- Le mot d'adresse 0b010110001 est-il dans le cache ? Si oui quelle est la valeur associée ? Si oui, la mémoire est-elle à jour ?
- Le mot d'adresse 0b111000001 est-il dans le cache ? Si oui quelle est la valeur associée ? Si oui, la mémoire est-elle à jour ?
- Le mot d'adresse 0b000000011 est-il dans le cache ? Si oui quelle est la valeur associée ? Si oui, la mémoire est-elle à jour ?
- Si la donnée 0x1234 située en mémoire à l'adresse 0x1BA était dans le cache, comment apparaitrait-elle ?

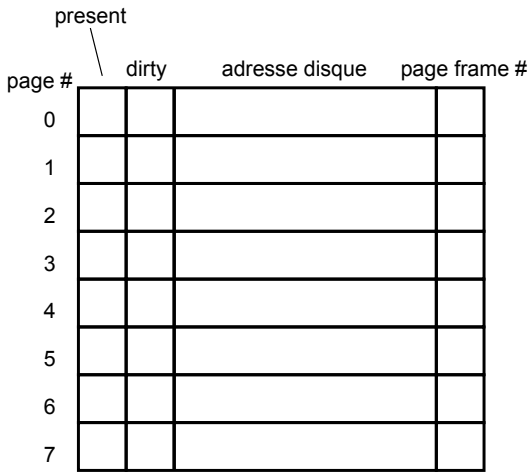
35. PROTOCOLE MESI DE COHERENCE DE CACHES

On suppose l'existence de 3 processeurs CPU1…CPU3 ayant chacun une cache line CL1…CL3 pour une même zone de données en mémoire, nommée A. On suppose également que cette zone A est accédée par les 3 CPU selon la séquence suivante : CPU1 lit A ; CPU2 lit A ; CPU1 modifie A ; CPU3 modifie A ; CPU1 lit A

- Décrire l'évolution des bits MESI de ces caches lines, depuis une situation initiale où ils sont tous invalides
- Indiquer tous les accès mémoire (lecture et écriture) effectués

36. MEMOIRE VIRTUELLE PAGINEE

On suppose l'existence d'une mémoire réelle de 4 pages frames, et d'une mémoire virtuelle de 8 pages, avec une table des pages de la forme suivante :



On suppose que le système utilise la stratégie FIFO lorsqu'une page frame est recopiée sur le disque. On suppose également que la traduction d'adresse est faite sur le schéma suivant :

1. Quelle est la taille de la mémoire virtuelle ? De la mémoire réelle ? Quelle est la taille des pages ?
2. On part d'un état initial où aucune page n'est présente et le processeur fait un premier accès mémoire, en écriture, à l'adresse virtuelle 0x0F46. Décrire l'évolution des bits: *present, dirty, page frame* dans la table des pages, et indiquer quels accès à la mémoire réelle et quels accès disque ont été effectués.
3. Faire de même en continuant avec les accès suivants du processeur : lecture en 0x0F46 ; écriture en 0x0800 ; écriture en 0x0F00 ; lecture en 0x0804 ; lecture en 0x0C00 ; lecture en 0x1800 ; lecture en 0x1C00 ; lecture en 0x0400

37. PIPELINE

On considère le programme suivant :

```
        setq    10, %r1
        clr     %r2
loop:   add     %r1, %r2, %r2
```

```
deccc       %r1
bne         loop
i6
i7
```

Indiquer dans le tableau suivant comment les instructions progressent dans le pipeline, initialement complètement rempli de bulles. On supposera que les branchements introduisent systématiquement un aléa de contrôle sans prédiction.

cycle	PC	Instr	Args	Calc	Mem	Result
1	o	o	o	o	o	o