Université Grenoble Alpes
MIAGE L3 – Architectures des Ordinateurs

Examen, première session
Décembre 202&, Durée 3h
(~1h00 Architectures des ordinateurs, ~1h00 Système, ~1h00 Réseau)
Sujet Architecture des ordinateurs sur 1 recto-verso.

I. Automates de Moore (barème indicatif : 5 points)

Rappelez la définition des automates de Moore. Dénombrez approximativement le nombre de circuits d'automates de Moore différents réalisables avec 1 bits d'entrée, 2 bits de mémoire et 3 bits de sortie. Expliquez votre calcul.

II. Racine carrée naïve. (barème indicatif : 8 points)

L'exercice consiste à transformer le programme ci-après en circuit. Au choix, vous pourrez le traduire sous forme de circuit à flot de données, ou sous forme de circuit PC-PO. Si vous choisissez la réalisation d'un circuit PC-PO, après avoir donné la « partie opérative » et l'automate formel de la « partie contrôle », vous pourrez réduire le dessin du circuit lui-même à quelques éléments représentatifs de chaque partie du circuit.

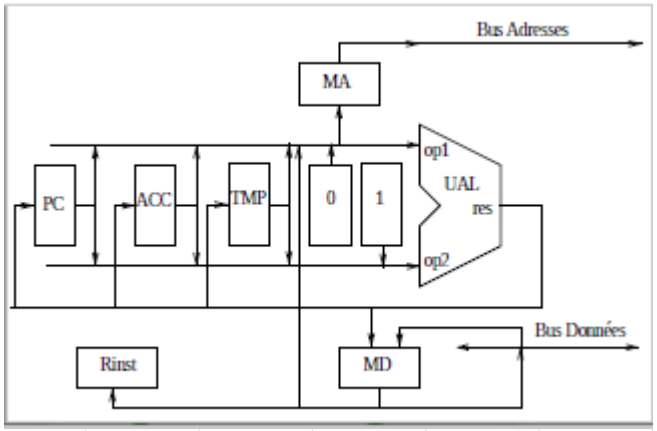
```
Entrée : X entier naturel sur n bits (prendre n=8 si nécessaire)
Sortie : R entier naturel sur n bits
Début
 | C ← 0, R ← 0
 | Tant que C < X :
 |    | C ← C + 2*R + 1
 |    | si C < X : R ← R+1  FinSi
 |    └ FinTantQue
 └ Fin
```

III. Machine à 5 instructions. (barème indicatif : 7 points)

L'exercice reprend une machine à accumulateur similaire à celle vue en cours. Elle comporte 5 instructions dont les codes sont sur 4 bits, la première n'a pas d'argument, les autres instructions nécessitent un opérande codé sur 4 bits :

- clr : initialise l'accumulateur avec la valeur 0, code 0001
- ld# vi : chargement d'une valeur vi dans l'accumulateur (adressage immédiat), code 0010
- st@ ad : rangement de l'accumulateur à l'adresse ad (adressage direct), code 0011
- ad@ ad : augmentation de l'accumulateur avec la valeur à l'adresse ad (adressage direct), code 0101
- jp@ ad : saut inconditionnel à l'adresse ad (adressage absolu), code 0100

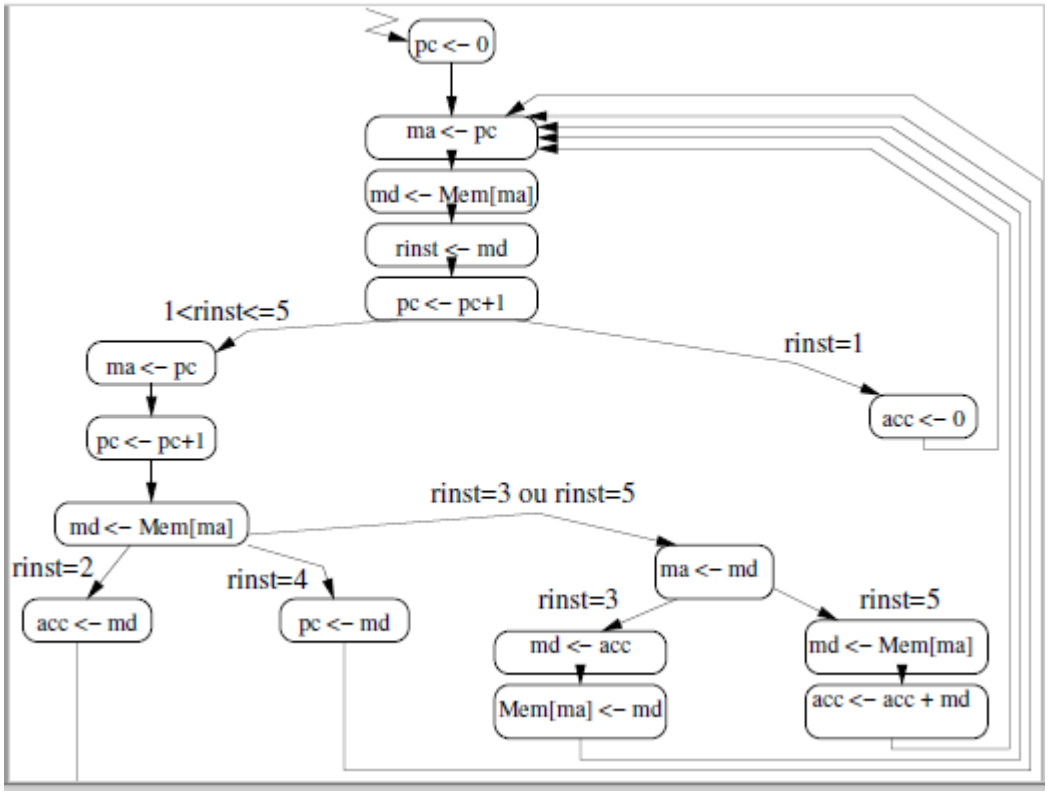
La partie opérative du processeur est donnée par le schéma ci-contre.
L'UAL est capable d'addition, soustraction et décalage et peut fournir en sortie les indicateurs Z et N.
TMP est un registre libre. MA et MD font le lien avec la mémoire. 0 et 1 sont des constantes.



La mémoire est donnée par :

@	Val						
0000	0010	0100	0101	1000	0100	1100	0000
0001	1011	0101	1111	1001	0100	1101	0000
0010	0011	0110	0011	1010	0000	1110	0000
0011	1111	0111	1110	1011	0000	1111	0000

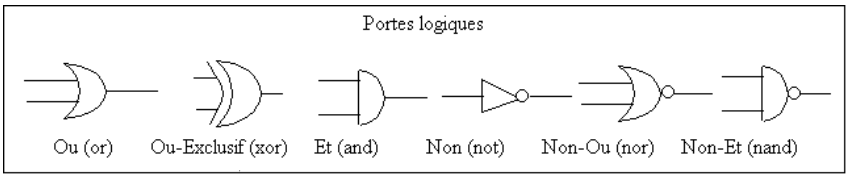
L'automate de contrôle est donné par le schéma ci-après.



Simulez le début de l'exécution du programme au niveau de détail donné par l'automate de contrôle (une dizaine de lignes de simulation).
Simulez une seconde fois l'exécution du programme au niveau de détail donné par le langage machine (une dizaine de lignes de simulation)

Complétez le langage machine par 3 instructions de multiplication M3, M5 et M7 qui respectivement multiplient l'accumulateur par 3, 5 et 7. Donnez une solution optimisant le nombre d'états. (vous pourrez utiliser librement le registre TMP).

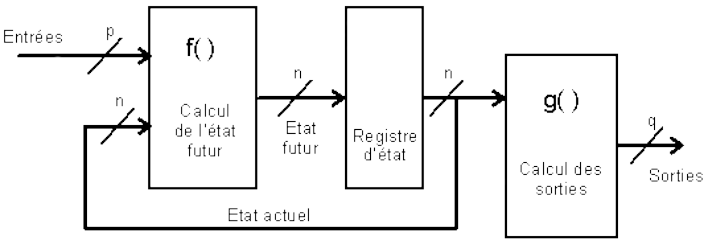
Rappels de cours



Portes logiques, ci-contre (et : *, ou : +)
Associative, Commutative, Distributive
de Morgan : $\overline{A+B} = \overline{A} * \overline{B}$, $\overline{A*B} = \overline{A} + \overline{B}$
Simplification : $A + \overline{A}C = A + C$

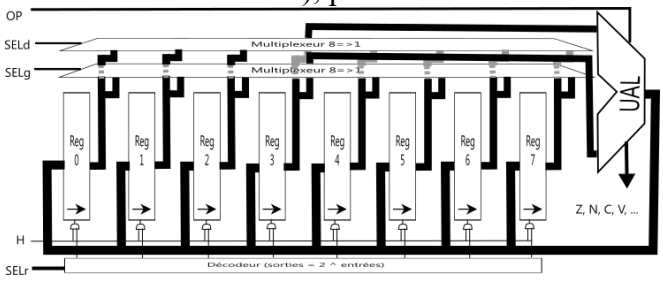
Bascule D : 2 entrées C et D, une sortie Q
affectation Q ← D sur front ; mémorisation Q_{prec} sinon

Automate (Moore, ci-contre)
dans les états : NomEtat ↑ Sorties
sur les transitions : Entrée

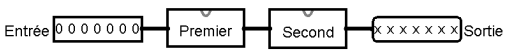


Circuit à flot de données (ci-contre), construction systématique par séquence, conditionnelle, boucle

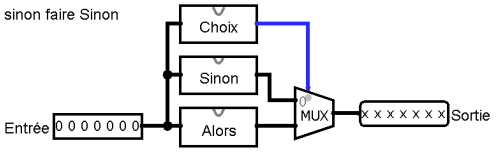
PC/PO, méthodologie : PO (prévoir registres, constantes et opérations, ci-dessous), PC (c'est un automate, faire le circuit de transition, de sortie et assembler le tout et état), puis assembler PC et PO.



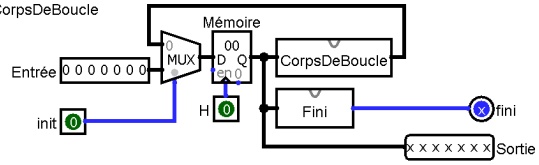
Séquence : Premier ; Second



Conditionnelle : Si (Choix) faire Alors sinon faire Sinon



Boucle : Tant que (pas Fini) faire CorpsDeBoucle



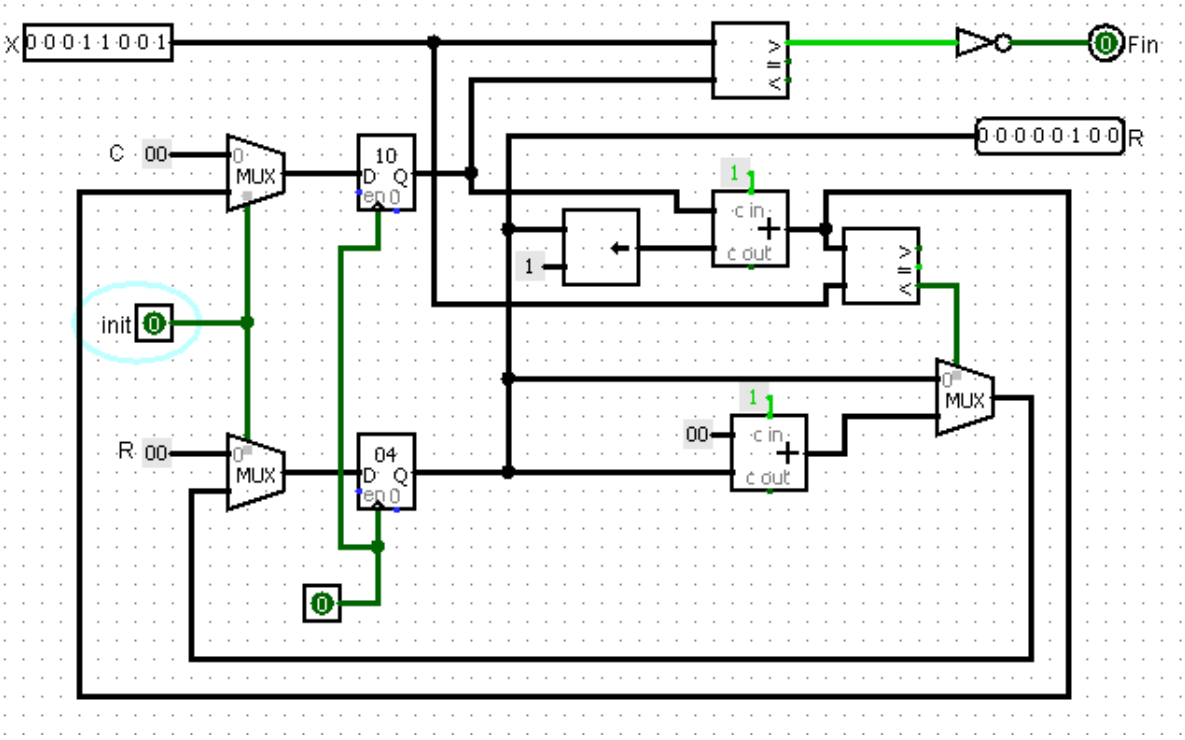
Eléments de correction :

Exercice 1. Automate de Moore

Définition des automates de Moore : Automate défini par un ensemble d'état, un vocabulaire d'entrée, un vocabulaire de sortie et 2 fonctions indépendante : la fonction de transition (etat,entree) ⇒ etat, et la fonction de sortie : etat ⇒ sortie.

Dénombrez le nombre de circuits d'automates de Moore avec 1bits d'entrée, 2 bits de mémoire et 3 bits de sortie : la table de vérité de la fonction de transition aura 2+1 colonnes en entrées, donc $2^3 = 8$ lignes et 2 colonnes en sortie, donc sera déterminée par 8 x 2 = 16 valeurs binaires, il y a donc 2^{16} fonctions de transitions différentes possibles (65536). Pour la fonction de sortie, la table a 2 colonne en entrée, donc 4 lignes et 3 colonnes en sortie donc sera déterminée 4 x 3 = 12 valeurs binaires, il y a donc 2^{12} fonctions de transitions différentes possibles (4096). Comme les fonctions de transition et de sorties sont indépendantes, il y a donc $2^{16} \times 2^{18} = 2^{28}$ (environ 1/4 de milliard).

Exercice 2. Racine Carrée



rem. : pour simplifier l'exercice, l'énoncé comportait une comparaison (`si C < X`), pour que le résultat soit juste en cas de racine carré entière, il faut utiliser (`si C ≤ X`). Le circuit proposé correspond à l'énoncé initial.

Exercice 3. Machine à 5 instructions

Exécution niveau automate (début) :

pc ⇐ 0 (0000) ; ma ⇐ pc (0000) ; md ⇐ Mem[ma] (0100) ; rinst ⇐ Md (0100) ; pc ⇐ pc+1 (0001) ;
ma ⇐ pc (0001) ; pc ⇐ pc +1 (0002) ; md ⇐ Mem[ma] (1011) ; acc ⇐ md (1011) ; retour ma ⇐ pc ; etc.

Exécution niveau assembleur (début) :

ld# 1011 (acc=1011) ; st@ 1111 (Mem[1111]=1011) ; add@ 1111 (acc=0110) ;
st@ 1110 (Mem[1110] = 0110) ; jp@ 0100 (pc = 0100) ; retour à add@ 1111 (acc=0001) ; etc

Ajout de 3 instructions M3, M5, M7 de multiplication de l'accumulateur par 3, 5 et 7 : les multiplications proposées sont similaires au clr (pas besoin de second argument), à la sortie du premier bloc, on peut après le premier pc ⇐ pc+1 ajouter une transition alternative commune à M3, M5, M7, mais uniquement pour ces 3 instructions vers un 1er état faisant TMP ⇐ ACC+ACC (ou Acc*2 par décalage si c'est possible), puis à la suite, un 2e état ACC ⇐ ACC + TMP qui a une transition qui boucle vers l'état de départ du fetch (ma ⇐ pc) pour M3, sinon a une transition vers un 3e état ACC ⇐ ACC + TMP qui a une transition qui boucle vers l'état de départ du fetch (ma ⇐ pc) pour M5, sinon, une transition vers un 4e état ACC ⇐ ACC + TMP qui a une transition qui boucle vers l'état de départ du fetch (ma ⇐ pc). La solution proposée ajoute 4 états.