université de BORDEAUX	**ANNÉE UNIVERSITAIRE 2025 – 2026** **SESSION 1 DE PRINTEMPS** **Parcours / Étape : LSTS / L2** **Code UE : 4TIN408U** **Épreuve :** Architecture des ordinateurs **Date :** 12 mai 2026 **Heure :** 09h00 **Durée :** 1h30 Documents : non autorisés Épreuve de M./Mme : F. Pellegrini	Collège Sciences et Technologies

N.B. : - Les réponses aux questions doivent être argumentées et aussi concises que possible.
- Le barème est donné à titre indicatif.
- Inscrivez votre numéro d'anonymat sur chacune des deux feuilles à joindre avant de les insérer dans votre copie.

Exercice 1 (30 points)

(1.1) (10 points)

Expliquer ce qu'est le complément à deux en arithmétique entière.
(10 lignes maximum)

(1.2) (20 points)

En utilisant des portes ET, OU et NON, construisez et dessinez un dispositif ayant les propriétés suivantes :

— le dispositif possèdera 3 lignes d'entrée et 2 lignes de sortie ;

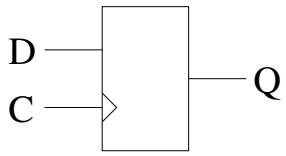
— les 3 lignes d'entrée représentent l'écriture, au moyen de la notation en complément à deux, de nombres $X = x_2x_1x_0$ en arithmétique entière signée ;

— les 2 lignes de sortie doivent fournir les valeurs suivantes :

Condition	S_0	S_1
$x = 0$	0	0
$x < 0$	0	1
$x > 1$	1	0
$x = 1$	1	1

Exercice 2 (60 points)

Les bascules D sont des circuits logiques à mémoire disposant de trois fils de contrôle : la sortie Q fournit en permanence l'état binaire (0 ou 1) mémorisé par le circuit, l'entrée D (pour « *data* ») permet de fournir au circuit la nouvelle valeur à mémoriser, celle-ci n'étant prise en compte que quand l'entrée C (pour « *clock* ») passe à l'état haut (front montant). On les représente schématiquement ainsi :



On dispose d'une horloge H fournissant un signal rectangulaire régulier de fréquence f.

(2.1) (10 points)

Comment câbler les fils D et Q d'une bascule pour qu'à chaque front montant de C la sortie Q de la bascule change d'état ? Dessinez le schéma correspondant.

(2.2) (10 points)

Tracer un chronogramme représentant l'évolution des états de H et de Q au cours du temps sur quatre cycles d'horloge ; on supposera que l'état initial de Q est 0. Au vu de ce schéma, que pouvez-vous dire du signal de Q par rapport à celui de H ?

T.S.V.P. ⟶

(2.3) (20 points)

Comment câbler deux bascules D, appelées D_0 et D_1, pour que leurs états de sortie Q_0 et Q_1 parcourent les quatre états 00, 01, 10, 11, dans cet ordre, en changeant d'état à chaque front montant de H ? Dessinez le schéma correspondant. Le signal Q_0 correspondra au bit de poids faible, et Q_1 au bit de poids fort.

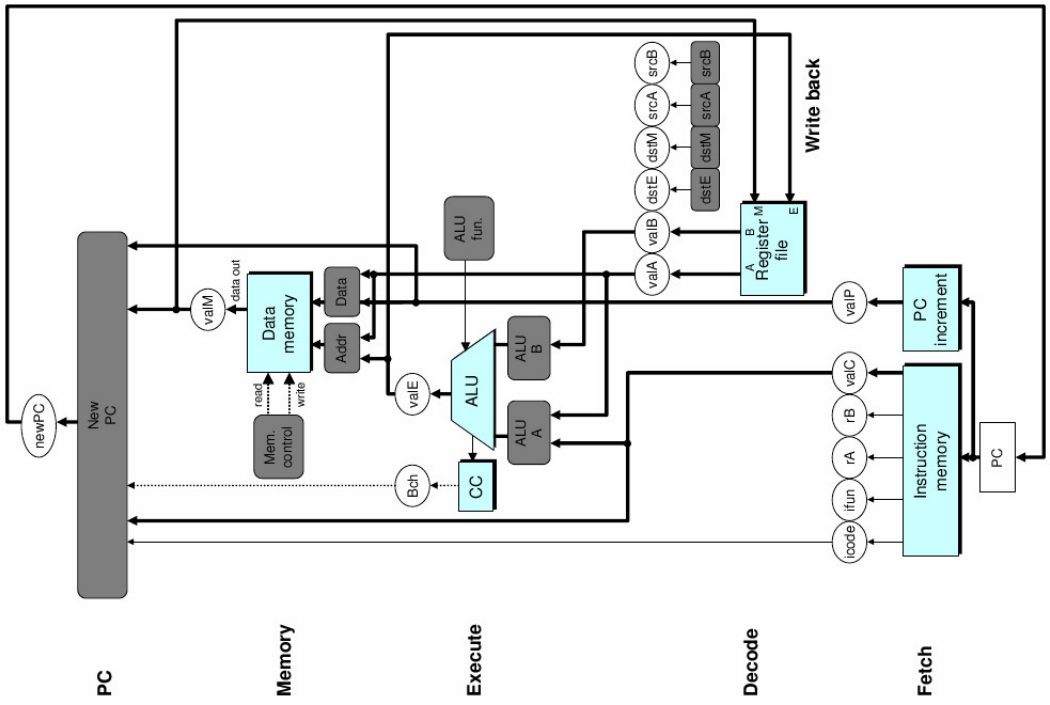
N.B. : Mettez en œuvre une solution qui ne crée pas de dérive de l'horloge !

(2.4) (20 points)

Réalisez le schema d'un circuit permettant de commander trois signaux de sortie afin de réaliser un « chenillard », prenant successivement les six états suivants : 000, 001, 011, 111, 110, 100, et ainsi de suite, et changeant d'état à chaque front montant de H. On étiquettera de S_0 à S_2 les sorties du chenillard, S_0 correspondant au bit de poids le plus faible des six représentations binaires ci-dessus.

Exercice 3 (70 points)

La figure ci-dessous est le schéma de l'architecture Y86 séquentielle.



Le tableau ci-dessous représente le fonctionnement des différents étages lors de l'exécution des trois instructions « `irmovl valC,rB` », « `popl rA` » et « `call valC` ».

Étage	irmovl valC,rB	popl rA	call valC
Fetch	icode:ifun = M_1[PC] rA:rB = M_1[PC+1] valC = M_4[PC+2] valP = PC + 6	icode:ifun = M_1[PC] rA:rB = M_1[PC+1] valP = PC + 2	icode:ifun = M_1[PC] valC = M_4[PC+1] valP = PC + 5
Decode		valA = R[%esp] valB = R[%esp]	valB = R[%esp]
Execute	ValE = 0 + valC	valE = valB + 4	valE = valB + (-4)
Memory		ValM = M_4[valA]	M_4[valE] = valP
Write back	R[rB] = valE	R[%esp] = valE R[rA] = valM	R[%esp] = valE
PC update	PC = valP	PC = valP	PC = valC

(3.1) (30 points)

En vous basant sur les informations précédentes, remplissez les tableaux correspondant aux instructions « `rrmovl rA,rB` », « `pushl rA` » et « `ret` ».

(3.2) (20 points)

Complétez le code HCL de la feuille jointe afin de prendre en compte les trois instructions que vous venez de définir.

(3.3) (10 points)

On souhaite ajouter au processeur l'instruction `xchgl rA,rB`, qui échange le contenu des deux registres arguments de l'instruction. Combien d'accès à la banque de registres sont-ils nécessaires pour mettre en œuvre cette instruction ?

(3.4) (10 points)

Pourquoi cette instruction ne peut-elle être mise en œuvre avec l'architecture présentée dans le schéma vu plus haut ? Justifiez votre réponse. De fait, quelles modifications proposez-vous au chemin de données du processeur pour supporter cette instruction ?

Exercice 4 (40 points)

On souhaite pipe-liner un circuit combinatoire. Pour cela, on a décomposé ce circuit en cinq blocs A à E de durées respectives 20, 90, 50, 40 et 60 ps. Ces blocs doivent être exécutés l'un après l'autre dans cet ordre, après quoi on charge un registre au prochain front d'horloge. La durée de chargement d'un tel registre est de 20 ps.

(4.1) (10 points)

Insérer un seul registre intermédiaire fournit un pipeline de profondeur 2. Où faut-il insérer ce registre pour obtenir un débit maximal ? Calculez la durée minimale du cycle d'horloge auquel peut être cadencé le pile-line, son débit et sa latence.

(4.2) (10 points)

Même question que ci-dessus en insérant deux registres intermédiaires au lieu d'un seul, pour obtenir un pipe-line de profondeur 3.

(4.3) (20 points)

Vous disposez maintenant d'autant de registres que vous en avez besoin. Quel est le pipeline de profondeur optimale ? Pourquoi ? Combien de registres utilise-t-il ? Comme précédemment, calculez la durée minimale de son cycle d'horloge, son débit et sa latence.

N.B. *: n'oubliez pas d'inscrire votre numéro d'anonymat sur la feuille avant de l'insérer dans votre copie.*

— Feuille à détacher et à rendre avec la copie —

Numéro d'anonymat :

```
################ Fetch Stage     ###################################

# Does fetched instruction require a regid byte?
bool need_regids =
icode in { OPL, IOPL, POPL, IRMOVL, RMMOVL, MRMOVL };

# Does fetched instruction require a constant word?
bool need_valC =
icode in { IRMOVL, RMMOVL, MRMOVL, JXX, CALL, IOPL };

################ Decode Stage    ###################################

## What register should be used as the A source?
int srcA = [
icode in { RMMOVL, OPL } : rA;
icode in { POPL } : RESP;
1 : RNONE; # Don't need register
];

## What register should be used as the B source?
int srcB = [
icode in { OPL, IOPL, RMMOVL, MRMOVL } : rB;
icode in { POPL, CALL } : RESP;
1 : RNONE;  # Don't need register
];

## What register should be used as the E destination?
int dstE = [
icode in { IRMOVL, OPL, IOPL } : rB;
icode in { POPL, CALL } : RESP;
1 : RNONE;  # Don't need register
];

## What register should be used as the M destination?
int dstM = [
icode in { MRMOVL, POPL } : rA;
1 : RNONE;  # Don't need register
];
```

T.S.V.P. ⟶

```
################ Execute Stage   ###################################

## Select input A to ALU
int aluA = [
icode in { OPL } : valA;
icode in { IRMOVL, RMMOVL, MRMOVL, IOPL } : valC;
icode in { CALL } : -4;
icode in { POPL } : 4;
# Other instructions don't need ALU
];

## Select input B to ALU
int aluB = [
icode in { RMMOVL, MRMOVL, OPL, IOPL, CALL, POPL } : valB;
icode in { IRMOVL } : 0;
# Other instructions don't need ALU
];

## Set the ALU function
int alufun = [
icode in { OPL, IOPL } : ifun;
1 : ALUADD;
];

## Should the condition codes be updated?
bool set_cc = icode in { OPL, IOPL };

################ Memory Stage    ###################################

## Set read control signal
bool mem_read = icode in { MRMOVL, POPL };

## Set write control signal
bool mem_write = icode in { RMMOVL, CALL };

## Select memory address
int mem_addr = [
icode in { RMMOVL, CALL, MRMOVL } : valE;
icode in { POPL } : valA;
# Other instructions don't need address
];

## Select memory input data
int mem_data = [
icode in { RMMOVL } : valA;
icode == CALL : valP;
# Default: Don't write anything
];

################ Program Counter Update ############################

## What address should instruction be fetched at

int new_pc = [
icode == CALL : valC;
icode == JXX && Bch : valC;
1 : valP;
];
```