Université de Bordeaux
Licence Informatique 2ème année

U.F. Informatique
Année 2020 – 2021

ARCHITECTURE DES ORDINATEURS

Devoir Surveillé 1

1 heure
sans documents

N.B. : - Travaillez calmement. Ne bâclez pas vos réponses : il vaut mieux traiter correctement moins de questions que tout faire de travers.
- Les réponses aux questions doivent être argumentées et aussi concises que possible.
- Le barème est donné à titre indicatif.

Exercice 1 (11 points)

Voici le texte d'un programme écrit en langage Y86. Tous les commentaires ayant malencontreusement été effacés, il vous faut partir de zéro pour comprendre ce que fait ce programme, en répondant aux questions ci-dessous.

Attention : les réponses qui, au lieu d'expliquer, paraphrasent simplement le code (telles que « on soustrait %edx à %ebx ») ne rapporteront aucun point.

```
0x000:                |           .pos    0
0x000: 308400020000 | main:     irmovl  pile,%esp
0x006: 500800010000 |           mrmovl  x,%eax
0x00c: a008         |           pushl   %eax
0x00e: 308004010000 |           irmovl  t,%eax
0x014: a008         |           pushl   %eax
0x016: 8022000000   |           call    mystere
0x01b: c08408000000 |           iaddl   8,%esp
0x021: 10           |           halt
                    |
0x022: a038         | mystere:  pushl   %ebx
0x024: 502408000000 |           mrmovl  8(%esp),%edx
0x02a: 50340c000000 |           mrmovl  12(%esp),%ebx
0x030: c18301000000 |           isubl   1,%ebx
0x036: 6033         |           addl    %ebx,%ebx
0x038: 6033         |           addl    %ebx,%ebx
0x03a: 6023         |           addl    %edx,%ebx
0x03c: 2030         | et1:      rrmovl  %ebx,%eax
0x03e: 6120         |           subl    %edx,%eax
0x040: 716e000000   |           jle     et2
0x045: 500300000000 |           mrmovl  (%ebx),%eax
0x04b: 501200000000 |           mrmovl  (%edx),%ecx
0x051: 401300000000 |           rmmovl  %ecx,(%ebx)
0x057: 400200000000 |           rmmovl  %eax,(%edx)
0x05d: c08204000000 |           iaddl   4,%edx
0x063: c18304000000 |           isubl   4,%ebx
0x069: 703c000000   |           jmp     et1
0x06e: b038         | et2:      popl    %ebx
0x070: 90           |           ret
                    |
0x100:              |           .pos    0x100
0x100: 04000000     | x:        .long   4
0x104: 04000000     | t:        .long   4
0x108: 02000000     |           .long   2
0x10c: 06000000     |           .long   6
0x110: 05000000     |           .long   5
0x114: 01000000     |           .long   1
                    |
0x200:              |           .pos    0x200
0x200: 00000000     | pile:     .long   0
```

(1.1) (1 point)
Quels sont les paramètres de la fonction `mystere` ? Expliquez quelle partie du code permet de répondre à cette question.

(1.2) (1 point)
Quel est le rôle de l'instruction d'adresse `0x022` ? Pourquoi est-elle nécessaire ?

(1.3) (2 points)
Quel est le rôle des instructions d'adresses `0x036` à `0x039` ? À quoi servent-elles, dans le cadre du bloc d'instructions allant des adresses `0x030` à `0x03b` ?

(1.4) (2 points)
De fait, quelle différence y a-t-il entre les valeurs initiales des registres `edx` et `ebx` ? Quelles sont leurs valeurs initiales dans le cadre de l'exécution de ce programme, et à quoi correspondent-elles ?

(1.5) (1 point)
À quoi le test de l'adresse `0x040` sert-il ? À quel moment ce branchement est-il pris ?

(1.6) (2 points)
Pour chacun des registres utilisés au sein de cette fonction, donnez son nom puis, en une unique phrase synthétique, son rôle au sein de la fonction.

(1.7) (1 point)
Donnez l'état de la mémoire, entre les adresses `0x100` et `0x113`, après l'exécution de la fonction `mystere`.

(1.8) (1 point)
Dites, en une phrase, ce que fait la fonction `mystere`.

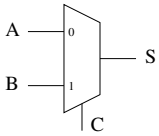
Exercice 2 (9 points)

On souhaite câbler le circuit « *IP Next* » d'un processeur, qui est le circuit permettant de calculer l'adresse de la prochaine instruction à exécuter.

(2.1) (1 point)

Écrivez l'équation logique d'un multiplexeur à deux entrées `A` et `B` sur un bit, et un bit de contrôle `C`, qui renvoie sur sa sortie `S` la valeur `A` si $\texttt{C} = 0$ et `B` si $\texttt{C} = 1$. Dessinez le schéma de câblage de ce multiplexeur, au moyen de portes logiques.

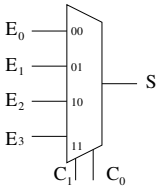
À partir de maintenant, et même si vous n'avez pas répondu à la question précédente, vous pouvez représenter un multiplexeur 1×2 *sous la forme du circuit ci-à côté, comprenant trois entrées* `A`*,* `B` *et* `C` *sur* 1 *bit, et une sortie* `S` *sur* 1 *bit.*



(2.2) (1 point)

En vous appuyant éventuellement sur le circuit précédent, concevez et dessinez le schéma de câblage d'un multiplexeur à deux bits de contrôle $\texttt{C}_0$ et $\texttt{C}_1$, codant un nombre i sur deux bits valant de 0 à 3 ($\texttt{C}_0$ étant le bit de poids faible), et qui renvoie sur sa sortie `S` la valeur de l'entrée $\texttt{E}_i$, parmi les quatre entrées $\texttt{E}_0$, $\texttt{E}_1$, $\texttt{E}_2$ et $\texttt{E}_3$.

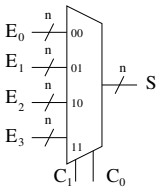
À partir de maintenant, et même si vous n'avez pas répondu aux questions précédentes, vous pouvez représenter un multiplexeur 1×4 *sous la forme du circuit ci-à côté, comprenant quatre entrées* E_0*,* E_1*,* E_2 *et* E_3 *sur* 1 *bit, une entrée de contrôle* $C = C_1 C_0$ *sur deux bits, et une sortie* `S` *sur* 1 *bit.*



(2.3) (1 point)

En vous appuyant sur le circuit précédent, dessinez le schéma de câblage d'un circuit multiplexeur à quatre entrées $\texttt{E}_0$, $\texttt{E}_1$, $\texttt{E}_2$ et $\texttt{E}_3$, chacune étant sur n bits, et deux bits de contrôle $\texttt{C}_1\texttt{C}_0$, et qui renvoie sur sa sortie `S`, sur n bits également, la valeur $\texttt{E}_i$.

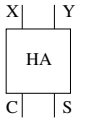
À partir de maintenant, et même si vous n'avez pas répondu aux questions précédentes, vous pouvez représenter un multiplexeur $n \times 4$ *sous la forme du circuit ci-à côté, comprenant quatre entrées* E_0*,* E_1*,* E_2 *et* E_3`A` *sur* n *bits, une entrée de contrôle* $C = C_1 C_0$ *sur deux bits, et une sortie* `S` *sur* n *bits.*



(2.4) (2 points)

Donnez les tables logiques, puis les fonctions logiques, d'un demi-additionneur `HA` à un bit, prenant en entrée deux valeurs `X` et `Y` sur un bit, et fournissant la somme `S` et la retenue `C` codant en binaire le résultat $\texttt{C}\,\texttt{S} = \texttt{X} + \texttt{Y}$.

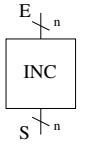
À partir de maintenant, et même si vous n'avez pas répondu aux questions précédentes, vous pouvez représenter un demi-additionneur HA sous la forme du circuit ci-à côté, comprenant deux entrées `X` *et* `Y` *sur* 1 *bit chacune, et deux sorties* `S` *et* `C` *également sur* 1 *bit.*



(2.5) (2 points)

En vous appuyant sur le circuit précédent, et sans utiliser d'additionneurs complets, dessinez le schéma de câblage d'un circuit incrémenteur `INC` sur n bits, à n entrées $\texttt{E}_i$ où $0 \leq i < n$, et n sorties $\texttt{S}_i$, telles que $\texttt{S} = (\texttt{E} + 1) \mod 2^n$.

À partir de maintenant, et même si vous n'avez pas répondu aux questions précédentes, vous pouvez représenter un incrémenteur `INC` *sous la forme du circuit ci-à côté, comprenant une entrée* `E` *sur* n *bits et une sortie* `S` *également sur* n *bits.*



(2.6) (2 points)

En vous appuyant sur les circuits précédents, construisez le circuit `IPN` doté des caractéristiques suivantes :

— une entrée `I` sur n bits, valant l'adresse courante ;

— une entrée `A` sur n bits, valant l'adresse de destination d'un branchement éventuel ;

— une entrée `J` sur 1 bit, indiquant si un branchement conditionnel doit être pris ;

— une entrée $\texttt{F} = \texttt{F}_1\texttt{F}_0$ sur deux bits, codant la fonction de branchement ;

— une sortie `N` sur n bits, égale à la valeur de l'adresse de la prochaine instruction.

Cette valeur `N` doit être égale à

— `I` si $\texttt{F} = 0$,

— $\texttt{I} + 1$ si $\texttt{F} = 1$,

— `A` si $\texttt{F} = 2$ et,

— dans le cas où $\texttt{F} = 3$, vaut `A` si $\texttt{J} = 1$ et $\texttt{I} + 1$ si $\texttt{J} = 0$.