Examen d'architecture des ordinateurs

EISC 1

11 juin 2019

Nom : ..... **Prénom :** .....

1 Représentation des données

Exercice 1. Conversion [2 pt]

Devinez la question et répondez !

Base 10	Représentation en complément à 2 sur 1 octet
72	
108	
	1110 0101
	0111 0001
−23	

Exercice 2. UTF-8 [4 pt]

On rappelle que la norme ASCII permet d'encoder jusqu'à 128 caractères différents sur 1 octet (le bit de poids fort vaut toujours 0). Voici une partie de la table ASCII :

Numéro ASCII (en base 10)	Caractère
10	saut de ligne
32	espace
33	!
63	?
65	A
66	B
97	a
98	b

Cependant, la norme ASCII ne permet pas d'encoder les caractères chinois, l'alphabet cyrillique, les émoticônes, ... Pour pallier ce problème, l'écrasante majorité des sites internet utilise la norme UTF-8. L'UTF-8 fonctionne ainsi : si un caractère appartient à la table ASCII alors il sera encodé par celle-ci sinon il sera encodé par les 3 octets suivant :

$$1110\,b_1b_2b_3b_4\ 10b_5b_6\,b_7b_8b_9b_{10}\ 10b_{11}b_{12}\,b_{13}b_{14}b_{15}b_{16}$$

où les 2 octets $b_1b_2b_3b_4\,b_5b_6b_7b_8\ b_9b_{10}b_{11}b_{12}\,b_{13}b_{14}b_{15}b_{16}$ sont le numéro Unicode du caractère. Voici une partie de la table Unicode :

Numéro Unicode (en base 2)	Caractère
1110 0111	ç
111 1101 1110	— (tiret cadratin)
10 0110 0011 1010	☺
100 1111 0110 0000	你
101 1001 0111 1101	好
	汉
	语

1. Décodez le message UTF-8 : 0101 0011 0110 0001 0110 1100 0111 0101 0111 0100

.....

.....

2. Encodez en UTF-8 le message : « 你好 !␣Hi ! »

.....

.....

.....

.....

3. Combien de symboles différents peuvent être encodés en UTF-8 ? Justifiez !

.....

.....

.....

.....

4. Quelle sera la taille en octets du dialogue ci-dessous après encodage en UTF-8 ? Justifiez !
   — 汉语?
   — ☺
   — Français␣ ?
   — Oui

.....

.....

.....

.....

2 Circuit

Exercice 3. Porte logique [4 pt]

On définit la porte BIDULE par la table de vérité suivante :

A	B	Sortie
0	0	1
0	1	0
1	0	0
1	1	0

1. Quel nom avez-vous envie de donner à cette porte ? Justifiez !

   .....

   .....

   .....

2. Construisez un circuit électronique à l'aide de transistors réalisant la porte BIDULE. Utilisez le moins de transistors possible.

3. Vous avez vu en TD que les portes NON et OU permettent de construire n'importe quelle fonction logique. Démontrez que la porte BIDULE est universelle (c'est-à-dire que n'importe quelle fonction logique peut être réalisée en utilisant uniquement la porte BIDULE).

   .....

   .....

   .....

   .....

Exercice 4. Unité arithmétique et logique [6 pt]

1. Qu'est-ce qu'un multiplexer ?

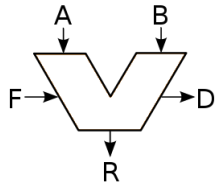
   .....

   .....

   .....

   .....

2. En utilisant uniquement des portes ET, OU et NON, construisez une UAL



telle que A, B, D, R soient codés sur 1 bit et F sur 2 bits. Votre UAL devra réaliser les opérations suivantes :

Valeur de F	Opération à réaliser
00	addition : $R = A + B$ et D servira de retenue si nécessaire
01	BIDULE (cf exercice 3) : $R =$ BIDULE(A, B)
10	multiplication : $R = A \times B$
11	comparaison : R vaut 1 si et seulement si $A < B$ et D vaut 1 si et seulement si $A = B$

3. Combien de transistors et de condensateurs utilise votre UAL ? Justifiez !

.....

.....

3 Programmation

Exercice 5. Triangle de Pascal [7 pt]

On rappelle qu'un code en NASM se structure ainsi

```
segment .data
segment .bss
segment .text

main :

mov ebx, 0
mov eax, 1
int 0x80
```

Comme en TP, vous pouvez appeler les fonctions *print_string* et *print_int*.

1. Écrivez une instruction permettant de déclarer un tableau *tab* de 30 octets. Dans quelle section devez vous placer cette instruction ?

   .....

   .....

2. Écrivez une suite d'instructions ou une fonction permettant d'initialiser toutes les cases du tableau *tab* à 0.

   .....

   .....

   .....

   .....

   .....

   .....

   .....

   .....

   .....

   .....

   .....

3. Écrivez une suite d'instructions ou une fonction permettant d'afficher le tableau *tab* case par case : si la case contient un nombre impair alors vous afficherez une croix "x" sinon vous afficherez une espace "␣". Par exemple, votre programme affichera comme ci-dessous le tableau [3;4;1;4;2;7;1;0]

   ```
   x x  xx
   ```

   .....

   .....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

4. On définit le triangle de Pascal par la suite :

$$T_{n,k} = \begin{cases} 1 & \text{si } k = n = 0 \\ T_{n-1,k-1} + T_{n-1,k} & \text{si } 0 \leq k \leq n \\ 0 & \text{sinon} \end{cases}$$

Voici les premiers termes de la suite :

n\ k	0	1	2	3	4
0	1	0	0	0	0
1	1	1	0	0	0
2	1	2	1	0	0
3	1	3	3	1	0
4	1	4	6	4	1

Écrivez un programme qui affiche, comme suit, le triangle de Pascal : le k^{e} caractère de la n^{e} ligne sera une croix "x" si le nombre $T_{n-1,k-1}$ est impair et une espace "␣" sinon. On placera dans le registre *edx*, au début de la fonction *main*, le nombre de lignes à afficher. Par exemple, si vous placez 16 dans le registre *edx* alors votre programme devra afficher :

```
x
xx
x x
xxxx
x   x
xx  xx
x x x x
xxxxxxxx
x       x
xx      xx
x x     x x
xxxx    xxxx
x   x   x   x
xx  xx  xx  xx
x x x x x x x x
xxxxxxxxxxxxxxxx
```

On pourra supposer que la valeur placée dans *edx* est inférieure à 32. Attention à la complexité de votre programme !

.....

.....

.....