GIF-1001 Ordinateurs: Structure et Applications
Automne 2016
Examen mi-session
26 octobre 2016
Durée: 170 minutes

Cet examen comporte 8 questions sur 18 pages (incluant celle-ci), comptabilisées sur un total de 100 points. L'examen compte pour 40% de la note totale pour la session. Assurez-vous d'avoir toutes les pages. Les règles suivantes s'appliquent:

- Vous avez droit à une feuille aide-mémoire 8.5×11 recto-verso écrite à la main;
- Écrivez vos réponses dans le cahier de réponses qui vous a été remis;
- L'annexe A contient une liste d'instructions ARM;
- L'annexe B contient la table ASCII.

La table ci-dessous indique la distribution des points pour chaque question.

Question:	1	2	3	4	5	6	7	8	Total
Points:	6	23	11	12	12	21	15	10	110

Bonne chance!

1. Répondez aux questions suivantes sur les ordinateurs.

   (a) (1 point) Quelle est la composante électronique qui à remplacé le tube à vide et qui à ouvert la porte aux microprocesseurs?

   Solution: Le transistor.

   (b) (2 points) Selon l'architecture de von Neumann, quelles sont les quatre (4) composantes principales d'un processeur?

   Solution: La mémoire, l'ALU, le CCU et l'équipement d'entrées-sorties. (0,5 pts par élément)

   (c) (1 point) Quelles sont les trois (3) grandes opérations du cycle d'instructions?

   Solution: Fetch, Decode et Execute. (0,5 s'il en manque 1, 0 s'il en manque 2 ou plus)

   (d) (2 points) Identifiez un périphérique qui est à la fois une entrée et une sortie pour l'ordinateur et expliquez pourquoi il s'agit d'une entrée et pourquoi il s'agit d'une sortie.

   Solution: Une imprimante, parce que l'ordinateur lui envoie quoi imprimer (sortie) et l'ordinateur reçoit le niveau d'encre (entrée).
   Un disque dur, parce que l'ordinateur lui envoie des données à enregistrer (sortie) et reçoit des données qui ont été sauvegardées auparavant (entrée).
   Un écran tactile, parce que l'ordinateur lui envoie ce qu'il doit afficher (sortie) et reçoit les coordonnées qui correspondent aux endroits où l'écran à été touché.
   (1 pt pour l'explication d'entrée, 1 pts pour l'explication de sortie)

2. Répondez aux questions suivantes sur la représentation des données dans un ordinateur.

   (a) (1 point) Combien de bits sont nécessaires pour stocker le jour du mois?

   Solution: Cinq (5) bits permettent de stocker 32 valeurs, c'est donc suffisant pour stocker 31 jours.

   (b) (3 points) Convertissez les nombres décimaux suivants en hexadécimal sur 20 bits: 4096, 512 et 170.

   Solution: 4096 = 0x01000, 512 = 0x00200, et 170 = 0x000AA.

   (c) (2 points) Additionnez les deux nombres suivants qui sont donnés en représentation complément-2 sur 12 bits et donnez la réponse en hexadécimal et en décimal: 0xFF5 et 0xFE3.

   Solution: 0xFD8, -40.

(d) (1 point) Indiquez si l'addition de la question précédente à provoqué une retenue (*carry*) et indiquez si elle a provoqué un débordement (*overflow*).

Solution: Elle a provoqué une retenue mais pas de débordement. (0,5 par élément de réponse)

(e) (1 point) Donnez deux nombres en hexadécimal sur 4 bits qui vont provoquer une retenue et un débordement lorsqu'on les additionne ensemble.

Solution: N'importe quels deux nombres entre 0x8 et 0xF inclusivement.

(f) (2 points) Lors d'une addition de deux nombres en complément-2, comment peut-on faire pour détecter un débordement? Donnez un exemple sur 5 bits.

Solution: On peut détecter un débordement lorsque le bit de signe est le même pour les deux opérandes en entrée, et qu'il change en sortie. Par exemple, $13 + 14$ est, en binaire, $01101_b + 01110_b = 11011_b$, soit -5. (2 pts si l'explication est entièrement correcte)

(g) (2 points) Donnez le nombre de bits à 1, le nombre de bits à 0 et le nombre total de bits dans le nombre suivant: 0xF0E1D2C3B4A59687.

Solution: 64 bits au total dont 32 à 0 et 32 à 1. (-1 si un des trois n'est pas correct, 0 si 2 erreurs)

(h) (2 points) Quel nombre est représenté par 0x42280000 encodé sur 32 bits avec IEEE 754? Rappel: la norme IEEE 754 emploie la formule

$$(\text{signe})1, \text{mantisse} \times 2^{(\text{exposant}-127)},$$

et les bits sont stockés selon la figure 1.

Figure 1: Convention IEEE-754 sur 32 bits.

Solution: 42

(i) (2 points) Quel nombre est représenté par 0x7FC42000 encodée sur 32 bits avec IEEE 754?

Solution: Ce n'est pas un nombre, not a number ou NaN.

(j) (2 points) Quel nombre est représenté par 0xff800000 encodé sur 32 bits avec IEEE 754?

Solution: $-\infty$.

(k) (3 points) Donnez la réprésentation IEEE 754 32 bits en hexadécimal de la fraction $\frac{7}{8}$.

Solution: La fraction s'exprime comme étant 7×2^{-3} soit en binaire $111_b \times 2^{-3} = 1,11_b \times 2^{-1}$ donc une mantisse de 11_b et un exposant de $-1 + 127 = 126$.

Cela nous donne la représentation hexadécimale 0x3F600000.

(1 point pour au moins 1 élément de trouvé, 2 points pour la bonne réponse)

(l) (2 points) Comment pouvez-vous représenter en mémoire sur 16 bits le nombre de pizzas que vous aller manger durant vos études universitaires avec une précision d'exactement $\frac{1}{32}$ de pizza? Dites quel est le nombre maximal de pizzas que vous pourrez représenter avec votre système.

Solution: Il s'agit d'utiliser un nombre point point fixe en réservant 5 bits de précision pour obtenir la précision de $\frac{1}{32}$ demandée. Cela nous laisse 11 bits pour la partie entière.

Le nombre maximal de pizzas pouvant être représenté avec ce système est 2047 et $\frac{31}{32}$ pizzas.

(1 point pour un système fonctionnel, 1 point pour le bon nombre maximal)

3. Pour les questions suivantes, considérez une mémoire pouvant stocker 128 octets (1 octet par adresse) dont une partie du contenu est illustré ci-dessous.

Adresse	Valeur
...	...
50_h	00_h
51_h	61_h
52_h	42_h
53_h	63_h
54_h	21_h
55_h	00_h
56_h	10_h
57_h	12_h
58_h	14_h
59_h	15_h
...	...
70_h	23_h
71_h	67_h
72_h	69_h
73_h	66_h
...	...

(a) (1 point) Quelle est la largeur minimale requise du bus d'adresse pour accéder à toutes les adresses de cette mémoire?

Solution: 7 bits.

(b) (1 point) Quelle est la largeur du bus de données utilisé par cette mémoire?

Solution: 8 bits.

(c) (1 point) Donnez le nombre total de bits pouvant être stockés dans cette mémoire.

Solution: 1024 bits.

(d) (2 points) Après avoir analysé un programme, vous avez déterminé qu'il met le mot de passe de l'utilisateur à l'adresse 81, et que le mot de passe a quatre (4) caractères. Quel est-il?

Solution: aBc! (0,5 point par caractère)

(e) (2 points) En faisant l'hypothèse d'un système petit boutiste (*little endian*) et qu'un nombre non-signé de 32 bits est en mémoire à l'adresse 86, quel est ce nombre en hexadécimal?

Solution: 0x15141210

(f) (2 points) En faisant l'hypothèse d'un système gros boutiste (*big endian*) et qu'un nombre non-signé de 32 bits est en mémoire à l'adresse 84, quel est ce nombre en hexadécimal?

Solution: 0x21001012

(g) (2 points) Quel type d'information vu dans le cadre du cours (entier, rationnel, caractère, chaîne de caractères, ...) est stocké en mémoire aux adresses 70_h à 73_h inclusivement?

Solution: On ne peut pas savoir. Peut-être une chaîne de caractères, peut-être un nombre entier 32 bits, peut-être 2 nombres entiers 16 bits, etc.

4. Un système de type “memory-mapped I/O” possède les caractéristiques suivantes:

   - un bus d’adresse de 21 bits, avec les 2 bits les plus significatifs (MSB) utilisés pour le décodeur d’adresse;
   - un bus de données de 16 bits;
   - une mémoire RAM où chaque octet possède une adresse différente;
   - trois autres périphériques sont branchés sur les bus;
   - il stocke les données en mémoire avec la convention “big endian”;
   - si on nomme les bits les plus significatifs (MSB) du bus d’adresse b_{20} et b_{19}, le décodeur sélectionne les périphériques de la façon suivante:

b_{20}	b_{19}	Périphérique activé
0	0	RAM
0	1	Périphérique 1
1	0	Périphérique 2
1	1	Périphérique 3

   (a) (2 points) Quelle est la taille maximale de la mémoire RAM? Écrivez votre réponse en kilo-octets (Ko).

   Solution: $21 - 2 = 19$ bits sont utilisés pour générer les adresses, et chaque octet possède une adresse différente. On a donc 2^{19} octets, soit $2^9 = 512$Ko.

   (b) (2 points) Quelle est la carte de la mémoire (*memory map*) de ce système?

   Solution: 0x000000–0x07FFFF: RAM
   0x080000–0x0FFFFF: P1
   0x100000–0x17FFFF: P2
   0x180000–0x1FFFFF: P3

Nous modifions le système afin qu’il puisse utiliser un bus de données de 32 bits au lieu de seulement 16. Le bus d’adresse ne change pas.

   (c) (2 points) Décrivez l’impact de ce changement sur la taille maximale de la mémoire RAM. Écrivez votre réponse en kilo-octets (Ko).

   Solution: La taille de la mémoire reste la même (512Ko) car le bus d’adresse ne change pas.

   (d) (2 points) Quelle est la carte de la mémoire (*memory map*) de ce système?

   Solution: La même:
   0x000000–0x07FFFF: RAM
   0x080000–0x0FFFFF: P1
   0x100000–0x17FFFF: P2
   0x180000–0x1FFFFF: P3

Finalement, nous remplaçons la mémoire RAM du système pour une mémoire qui attribue une adresse différente à chaque mot de 16 bits au lieu de seulement 8. Le système possède maintenant un bus de données de 32 bits, et cette nouvelle mémoire.

(e) (2 points) Décrivez l'impact de ce changement sur la taille maximale de la mémoire RAM. Écrivez votre réponse en kilo-octets (Ko).

Solution: La taille de la mémoire double (1024Ko) car la taille des mots en mémoire RAM double.

(f) (2 points) Quelle est la carte de la mémoire (*memory map*) de ce système?

Solution: La même:
0x000000–0x07FFFF: RAM
0x080000–0x0FFFFF: P1
0x100000–0x17FFFF: P2
0x180000–0x1FFFFF: P3

5. Répondez aux questions suivantes portant sur le micro-processeur du simulateur du travail pratique 1. Un rappel du jeu d'instructions est fourni:

Toutes les instructions du microprocesseur sont sur 16 bits et se décomposent comme suit:

- Bits 15 à 12: Opcode de l'instruction
- Bits 11 à 8: Registre utilisé comme premier paramètre.
- Bits 7 à 0: Registre ou constante utilisés comme deuxième paramètre

Le jeu d'instruction supporte les instructions suivantes où Rd est le registre destination, Rs le registre source et Rc le registre de condition:

Mnémonique	**Opcode**	**Description**
MOV Rd Rs	0000	Écriture de la valeur du registre Rs dans le registre Rd
MOV Rd Const	0100	Écriture d'une constante dans le registre Rd
ADD Rd Rs	0001	Addition des valeurs des registres Rd et Rs et insertion du résultat dans le registre Rd
ADD Rd Const	0101	Addition de la valeur du registre Rd avec une constante et insertion du résultat dans Rd
SUB Rd Rs	0010	Soustraction de la valeur Rs à l'intérieur de registre Rd.
SUB Rd Const	0110	Soustraction d'une constante à l'intérieur du registre Rd
LDR Rd [Rs]	1000	Chargement d'une valeur se trouvant à l'adresse Rs de l'ordinateur dans un registre.
STR Rd [Rs]	1001	Écriture de la valeur d'un registre à l'adresse Rs de l'ordinateur.
JZE Rc Const	1111	Saut à l'instruction située à l'adresse identifiée par la constante, mais seulement si Rc = 0 (sinon, cette instruction n'a aucun effet).
JZE Rc Rs	1011	Saut à l'instruction située à l'adresse Rs seulement si Rc = 0 (sinon, cette instruction n'a aucun effet).

Table 1: Jeu d'instructions du microprocesseur

(a) (5 points) Traduisez le programme suivant en binaire, et écrivez votre réponse en hexadécimal. Les numéros de ligne sont indiqués à gauche.

```
1 MOV R3 #32
2 LDR R2 [R3]
3 ADD R2 R2
4 ADD R2 #1
5 STR R2 [R3]
```

Solution:
0x4320
0x8203
0x1202
0x5201
0x9203
(1 point par instruction)

(b) (7 points) Soit le programme suivant. Pour chaque ligne, on indique l'adresse (qui commence à `0x0`), suivie de l'instruction en format binaire. Les numéros de ligne sont indiqués à gauche. Expliquez ce que fait ce programme en une phrase.

```
1 0x0   0x4000
2 0x1   0x8000
```

```
0x2   0x0100
0x3   0x4200
0x4   0x6101
0x5   0x1200
0x6   0xF109
0x7   0x4300
0x8   0xF304
0x9   0x9201
```

Solution: Il charge la valeur en mémoire à l'adresse 0x0, la met au carré, puis l'écrit en mémoire à cette même adresse. (1 point pour charge la valeur à l'adresse 0, 5 point pour la met au carré, 1 point pour écrit le résultat à l'adresse 0)

6. Répondez aux questions suivantes, portant sur l'assembleur ARM.

   (a) (1 point) Quel est le programme qui permet de créer un programme qui contient à la fois de l'assembleur et du code d'un autre langage?

   Solution: L'éditeur de liens.

   (b) (1 point) Quelle est la particularité des instructions de type SIMD?

   Solution: Ces instructions traitent plusieurs données avec une seule instruction.

   (c) (1 point) Auquel des deux grands types de jeu d'instructions le jeu d'instructions ARM appartient-il?

   Solution: RISC.

   (d) (3 points) Quels sont les registres spéciaux utilisés dans l'architecture ARM et quelles sont leurs fonctions?

   Solution:

   - PC (Program counter) Adresse de l'instruction à lire
   - LP (Link Register) Adresse de retour de fonction
   - SP (Stack Pointer) Adresse du sommet de la pile

   (0,5 par registre et par fonction)

   (e) (5 points) Considérez le code suivant (les numéros de ligne sont indiqués à gauche):

```
1   MOV SP, #0x3
2   MOV LR, PC
3 test
4   BX LR
5   BL test
6   CMP LR, PC
7   BLT test
8   MOV R0, #0x1
9   B test
10 fin
11  B fin
```

   Indiquez l'ordre des instructions exécutées par le microprocesseur en utilisant leur numéro de ligne correspondant.

   Solution: 1–2–4–5–4–6–7–4–6–7–4–6–7–... (2 point pour les 3 premiers, 3 points pour le reste)

   (f) (5 points) Écrivez du code assembleur qui place dans `R0` la valeur 1 si `R1` est pair ou -1 si `R1` est impair.

Solution: Par exemple:

```
TST R1, #1
MOVEQ R0, #1
MOVNE R0, #-1
```

(5 points pour une solution fonctionnelle)

(g) (5 points) Écrivez du code assembleur qui calcule la factorielle d'un nombre placé dans `R0`. En d'autres mots, implémentez le pseudo-code suivant:

```
R1 ← 1 ;
while R0 ≠ 0 do
    R1 ← R1 × R0 ;
    R0 ← R0 − 1 ;
end
```

Solution: Par exemple:

```
      MOV R1, #1

    boucle
      CMP R0, #0
      BEQ fin

      MUL R1, R1, R0
      SUB R0, R0, #1
      B boucle

    fin
```

(5 points pour un solution fonctionnelle et 2 points seulement pour une solution presque fonctionnelle)

7. Répondez aux questions portant sur le code assembleur ARM suivant (les numéros de ligne sont indiqués à gauche):

```
1   B       main
2
3  tableau DC32 0x10, 0x42, 0xA4, 0xA0, 0x32, 0x05, 0x45, 0x02, 0x00
4
5  main
6    LDR SP, =maPile
7    ADD SP, SP, #64
8
9    LDR R0, =tableau
10   BL fonctionMystere
11   MOV R5, R0
12
13   B main
14
15 fonctionMystere
16   PUSH {R1, R2, LR}
17   LDR R1, [R0], #4
18   MOV R2, R1
19
20 debut
21   CMP R1, #0x00
22   BEQ fin
23
24   CMP R2, R1
25   MOVLT R2, R1
26   LDR R1, [R0], #4
27   B debut
28
29 fin
30   MOV R0, R2
31   POP {R1, R2, LR}
32   BX LR
33
34 maPile DS32  16
```

(a) (1 point) Pourquoi utilise-t-on l'instruction `BL` et non `B` à la ligne 10?

Solution: Pour enregistrer l'adresse de retour dans le registre `LR`.

(b) (2 points) Pourquoi utilise-t-on les instructions `PUSH` et `POP` aux lignes 16 et 31?

Solution: Pour sauvegarder les registres que la fonction va utiliser, et les restaurer à leur valeur originale à la fin de la fonction. (1 point pour l'explication du push, 1 point pour l'explication du pop)

(c) (3 points) Que fait l'instruction `LDR R1, [R0], #4` à la ligne 17?

Solution: Elle copie dans `R1` le contenu de la mémoire à l'adresse contenue dans `R0`, puis incrémente `R0` de 4. (1 point pour 1 element de réponse, 3 point pour la solution complète.)

(d) (4 points) Quelle est la valeur de `R5` après l'exécution de l'instruction `MOV R5, R0` à la ligne 11?

Solution: 0xA4

(e) (5 points) Décrivez succinctement ce que fait la fonction `fonctionMystere`. De plus, indiquez ses arguments, la façon dont elle les obtient du programme principal, et comment elle retourne sa valeur de sortie.

Solution: Elle calcule la valeur maximale d'un tableau de nombres. Le tableau est parcouru jusqu'à ce que la valeur 0 soit rencontrée. L'adresse du tableau est fournie en entrée via le registre `R0`, et la valeur maximale est retournée via le registre `R0` également. (3 points pour ce qu'elle fait, 1 point pour les entrées, 1 point pour les sorties)

8. Répondez aux questions suivantes.

   (a) (1 point) Donnez un exemple d'un nombre hexadécimal qu'on ne peut pas représenter en décimal.

   Solution: Il n'y en a pas.

   (b) (1 point) Que désigne l'acronyme RISC? Donnez un exemple de famille de microprocesseurs lui correspondant.

   Solution: Reduced Instruction Set Computer, ARM.

   (c) (1 point) Que désigne l'acronyme CISC? Donnez un exemple de famille de microprocesseurs lui correspondant.

   Solution: Complex Instruction Set Computer, x86.

   (d) (1 point) Quel sous-sytème détermine la première micro-instruction à exécuter pour une certaine instruction?

   Solution: La mémoire de décodage.

   (e) (1 point) Quel est le nombre maximal d'instructions dans un jeu d'instructions?

   Solution: Il n'y en a pas. Cela dépend de la longueur des opcodes dans les instructions.

   (f) (1 point) Donnez deux exemples de mnémoniques d'instructions de gestion de la séquence d'instructions.

   Solution: BL et B.

   (g) (1 point) En assembleur ARM, donnez l'instruction qui permet de placer l'adresse de `maVariable` dans le registre R5.

   Solution: `LDR R5, =maVariable`

   (h) (1 point) Dans une architecture de type "memory-mapped I/O", quelle composante est responsable d'activer la broche d'activation (*enable*) des périphériques?

   Solution: Le décodeur d'adresse.

   (i) (1 point) Quelle composante est responsable de configurer l'ALU dans le bon mode?

   Solution: La mémoire de micro-instructions.

   (j) (1 point) Expliquez la différence entre les mémoires volatiles et non-volatiles?

Solution: Les mémoire volatiles perdent leur contenu lorsqu'elles perdent leur alimentation.

A Annexe: Instructions ARM et codes de conditions

Instruction	Description
`ADD Rd, Rs, Op1`	`Rd` ← `Rs + Op1`
`AND Rd, Rs, Op1`	`Rd` ← `Rs AND Op1`
`ASR Rd, Rs, #imm`	`Rd` ← `Rs` / $2^{\texttt{imm}}$
`Bcc Offset`	`PC` ← `PC + Offset`, si `cc` est rencontré
`BLcc Offset`	Comme `B`, `LR` ← adresse de l'instruction suivante
`CMP Rs, Op1`	Change les drapeaux comme `Rs - Op1`
`LDR Rd, [Rs, Op2]`	`Rd` ← `Mem[Rs + Op2]`
`LDR Rd, [Rs], Op2`	`Rd` ← `Mem[Rs]`, `Rs` ← `Rs + Op2`
`LDR Rd, [Rs, Op2]!`	`Rs` ← `Rs + Op2`, `Rd` ← `Mem[Rs]`
`LSL Rd, Rs, #imm`	`Rd` ← `Rs` × $2^{\texttt{imm}}$
`MUL Rd, Rs, Op1`	`Rd` ← `Rs` × `Op1`
`MVN Rd, Op1`	`Rd` ← `!Op1` (inverse les bits)
`POP {Reg List}`	Récupère la liste de registres de la pile
`PUSH {Reg List}`	Met la liste de registres sur la pile
`STR Rd, [Rs, Op2]`	`Mem[Rs + Op2]` ← `Rd`
`STR Rd, [Rs], Op2`	`Mem[Rs]` ← `Rd`, `Rs` ← `Rs + Op2`
`STR Rd, [Rs, Op2]!`	`Rs` ← `Rs + Op2`, `Mem[Rs]` ← `Rd`
`SUB Rd, Rs, Op1`	`Rd` ← `Rs - Op1`

Table 2: Instructions ARM. `Op1` dénote une opérande de type 1, et `Op2` une opérande de type 2.

Code	Condition	Code	Condition
`CS`	Retenue (carry)	`CC`	Pas de retenue
`EQ`	Égalité	`NE`	Inégalité
`VS`	Débordement	`VC`	Pas de débordement
`GT`	Plus grand	`LT`	Plus petit
`GE`	Plus grand ou égal	`LE`	Plus petit ou égal
`PL`	Positif	`MI`	Négatif

Table 3: Codes de condition.

B Annexe: Table ASCII

Dec	Hx	Oct	Char	Dec	Hx	Oct	Char	Dec	Hx	Oct	Char
0	0	000	NUL	43	2B	053	+	86	56	126	V
1	1	001	SOH	44	2C	054	,	87	57	127	W
2	2	002	STX	45	2D	055	-	88	58	130	X
3	3	003	ETX	46	2E	056	.	89	59	131	Y
4	4	004	EOT	47	2F	057	/	90	5A	132	Z
5	5	005	ENQ	48	30	060	0	91	5B	133	[
6	6	006	ACK	49	31	061	1	92	5C	134	\
7	7	007	BEL	50	32	062	2	93	5D	135	]
8	8	010	BS	51	33	063	3	94	5E	136	^
9	9	011	TAB	52	34	064	4	95	5F	137	_
10	A	012	LF	53	35	065	5	96	60	140	‘
11	B	013	VT	54	36	066	6	97	61	141	a
12	C	014	FF	55	37	067	7	98	62	142	b
13	D	015	CR	56	38	070	8	99	63	143	c
14	E	016	SO	57	39	071	9	100	64	144	d
15	F	017	SI	58	3A	072	:	101	65	145	e
16	10	020	DLE	59	3B	073	;	102	66	146	f
17	11	021	DC1	60	3C	074	¡	103	67	147	g
18	12	022	DC2	61	3D	075	=	104	68	150	h
19	13	023	DC3	62	3E	076	¿	105	69	151	i
20	14	024	DC4	63	3F	077	?	106	6A	152	j
21	15	025	NAK	64	40	100	@	107	6B	153	k
22	16	026	SYN	65	41	101	A	108	6C	154	l
23	17	027	ETB	66	42	102	B	109	6D	155	m
24	18	030	CAN	67	43	103	C	110	6E	156	n
25	19	031	EM	68	44	104	D	111	6F	157	o
26	1A	032	SUB	69	45	105	E	112	70	160	p
27	1B	033	ESC	70	46	106	F	113	71	161	q
28	1C	034	FS	71	47	107	G	114	72	162	r
29	1D	035	GS	72	48	110	H	115	73	163	s
30	1E	036	RS	73	49	111	I	116	74	164	t
31	1F	037	US	74	4A	112	J	117	75	165	u
32	20	040	Space	75	4B	113	K	118	76	166	v
33	21	041	!	76	4C	114	L	119	77	167	w
34	22	042	”	77	4D	115	M	120	78	170	x
35	23	043	#	78	4E	116	N	121	79	171	y
36	24	044	$	79	4F	117	O	122	7A	172	z
37	25	045	%	80	50	120	P	123	7B	173	{
38	26	046	&	81	51	121	Q	124	7C	174	
39	27	047	’	82	52	122	R	125	7D	175	}
40	28	050	(	83	53	123	S	126	7E	176	~
41	29	051	)	84	54	124	T	127	7F	177	DEL
42	2A	052	*	85	55	125	U				