UNIVERSITE GRENOBLE ALPES
Licence de Sciences et technologie - Parcours MIN, INF

Contrôle Continu UE INF451 : Architectures des ordinateurs

Mars 2020, durée 1 h 30
Document : 1 A4 R/V personnel manuscrit autorisé ; calculettes et téléphones portables interdits.
La plupart des questions sont indépendantes, si vous avez du mal avec l'une, passez à la suivante.
Le barème, sur 21 !, est donné à titre indicatif.

1 Numération en complément à 2 (6 points)

Pour cet exercice, **donnez tous les détails de calcul et justifiez vos réponses** (poser chaque opération avec les opérandes, les retenues, le résultat apparent et les indicateurs).

Questions :

(a) Déterminer le nombre de bits minimum pour représenter chacun des nombres suivants puis donner la représentation binaire et hexadécimale de ces entiers relatifs avec le nombre de bits choisi précédemment (codage en complément à 2) : $(+1337)_{10}$, $(-1337)_{10}$. **(2 points)**

Réponse : Il faut l'intervalle [-2048,+2047] donc 12 bits

$(+1337)_{10} = (0101\ 0011\ 1001)_2 = (5\ 3\ 9)_{16}$

Complément à 1 : $(1010\ 1100\ 0110)_2 = (A\ C\ 6)_{16}$
Complément à 2 : $(1010\ 1100\ 0111)_2 = (A\ C\ 7)_{16}$
D'où $(-1337)_{10} = (1010\ 1100\ 0111)_2$

(b) Donner les valeurs décimales des 2 entiers relatifs suivants codés sur 16 bits en complément à 2 : $(0000\ 1100\ 0100\ 0001)_2$ et $(FF8D)_{16}$. **(2 points)**

Réponse : $(0000\ 1100\ 0100\ 0001)_2$ donne en décimal 3137
$(FF8D)_{16}$ est négatif donc il faut le complément à 2 pour avoir sa valeur entière : $(FF8D)_{16} = (1111\ 1111\ 1000\ 1101)_2$
Complément à 1 : $(0000\ 0000\ 0111\ 0010)_2$
Complément à 2 : $(0000\ 0000\ 0111\ 0011)_2 = (64 + 32 + 16 + 2 + 1)_{10}$
D'où $(FF8D)_{16} = \S(-115)_{10}$

(c) Poser chacune des opérations suivantes sur 1 octet, donner le résultat binaire et apparent (codage complément à 2 sur 1 octet) et la valeur des indicateurs (Z, N, C (ou E pour la vraie soustraction) et V) : $(+47)_{10} + (-77)_{10}$, $(-57)_{10} - (+77)_{10}$ (vraie soustraction) et $(-47)_{10} - (+57)_{10}$ (soustraction par addition du complémentaire) . **(2 points)**

```
+47 = 0010 1111
Complement a 1 : 1101 0000
-47 = 1101 0001

+57 = 0011 1001
```

```
complement a 1 : 1100 0110
-57 = 1100 0111

+77 = 0100 1101
complement a 1 : 1011 0010
-77 = 1011 0011


+47 + -77
   00111 111
    0010 1111
    1011 0011
    1110 0010
C=0,N=1,V=0,Z=0

-57 - +77  (vraie soustraction)
   01111 000
    1100 0111
    0100 1101
    0111 1010

E=0,N=0,V=1,Z=0

-47 - +57
   11000 111
    1101 0001
    1100 0111
    1001 1000
C=1,N=1,V=0,Z=0
```

2 Codage de Cesar (ARM)

voir Caseine et YouTube.
https ://moodle.caseine.org/mod/vpl/view.php ?id=37526
https ://www.youtube.com/watch ?v=tHMYTPqElNs

3 Codage base64 (4 points)

« En informatique, base64 est un codage de l'information utilisant 64 caractères, choisis pour être disponibles sur la majorité des systèmes. Défini en tant qu'encodage MIME dans le RFC 2045, il est principalement utilisé pour la transmission de messages (courrier électronique et forums Usenet) sur Internet. Il est par ailleurs défini en propre dans le RFC 4648.

Description : Un alphabet de 65 caractères est utilisé pour permettre la représentation de 6 bits par un caractère. Le 65e caractère (signe « = ») n'est utilisé qu'en complément final dans le processus de codage d'un message.

Ce processus de codage consiste à coder chaque groupe de 24 bits successifs de données par une chaîne de 4 caractères. On procède de gauche à droite, en concaténant 3 octets pour créer un seul groupement de 24 bits (8 bits par octet). Ils sont alors séparés en 4 nombres de seulement 6 bits (qui en binaire ne permettent que 64 combinaisons). Chacune des 4 valeurs est enfin représentée (codée) par un caractère de l'alphabet retenu. (Table ci-dessous.)

Ainsi 3 octets quelconques sont remplacés par 4 caractères, choisis pour être compatibles avec tous les systèmes existants.

Chaque valeur (chaque groupe de 6 bits) est utilisée comme index dans la table ci-dessous. Le caractère correspondant est indiqué dans la colonne codage. »

Valeur	Codage	Valeur	Codage	Valeur	Codage	Valeur	Codage
0	000000 A	17	010001 R	34	100010 i	51	110011 z
1	000001 B	18	010010 S	35	100011 j	52	110100 0
2	000010 C	19	010011 T	36	100100 k	53	110101 1
3	000011 D	20	010100 U	37	100101 l	54	110110 2
4	000100 E	21	010101 V	38	100110 m	55	110111 3
5	000101 F	22	010110 W	39	100111 n	56	111000 4
6	000110 G	23	010111 X	40	101000 o	57	111001 5
7	000111 H	24	011000 Y	41	101001 p	58	111010 6
8	001000 I	25	011001 Z	42	101010 q	59	111011 7
9	001001 J	26	011010 a	43	101011 r	60	111100 8
10	001010 K	27	011011 b	44	101100 s	61	111101 9
11	001011 L	28	011100 c	45	101101 t	62	111110 +
12	001100 M	29	011101 d	46	101110 u	63	111111 /
13	001101 N	30	011110 e	47	101111 v		
14	001110 O	31	011111 f	48	110000 w	(complément)	=
15	001111 P	32	100000 g	49	110001 x		
16	010000 Q	33	100001 h	50	110010 y		

extrait de l'article base64 issu de Wikipedia.

Partie A. Codage (3 points).

Décrire par un schéma la méthode de codage base64.

Effectuer le codage des 3 premiers octets du premier exemple à suivre et des 6 octets de l'exemple suivant :

— Un fichier texte (ascii) commençant par "Il etait une fois ..."

Réponse :

Les 3 premiers octets : "Il " : 0x49 0x6C 0x20

Les 3 premiers octets en binaire : 0100 1001 0110 1100 0010 0000

Les 3 premiers octets en binaire redécoupage en paquets de 6 : 010010 010110 110000 100000

Les 3 premiers octets en base64 : SWwg

— Un fichier binaire commençant par les mots 32 bits 0xABCDEF01 0x23456789 ...

Réponse :

Les 6 premiers octets : AB CD EF 01 23 45

Les 6 premiers octets en binaire : 1010 1011 1100 1101 1110 1111 0000 0001 0010 0011 0100 0101

Les 6 premiers octets en binaire redécoupage en paquets de 6 : 101010 111100 110111 101111 000000 010010 001101 000101

Les 6 premiers octets en base64 : q83vASNF

Partie B. Analyse (1 point).
Répondez en quelques lignes aux questions suivantes :

— Quelles sont les avantages et inconvénients d'un codage base64 ?

Réponse :

- Avantage : Codage en ASCII donc portable (pas de problème compatibilité car l'ASCII est standard), beaucoup d'outils de mails (par exemple) utilise uniquement l'ASCII
- Inconvénient : Le code est 1/3 plus gros que l'original

— Si la séquence de données à coder n'a pas un nombre d'octets multiple de 3, est-ce un problème ?

Réponse : Le problème est que l'on code par paquets de 3 octets, dont il faut un nombre de donnée multiple de 3.

Si ce n'est pas le cas : il nous reste à la fin soit 1 ou 2 octets à coder.

- Si il nous en reste 1 : on code « normalement » les 6 premiers bits, on complète les 2 bits restants avec 4 zéros (à droite), on code ce nouveau paquet de 6 et enfin on termine avec deux caractères spéciaux « = » (ce qui distinguera ce cas particulier lors du décodage)
- Si il nous en reste 2 : on code « normalement » les 12 premiers bits (c-à-d en deux paquets de 6), on complète les 4 bits restants avec deux zéros (à droite) et enfin on termine avec un caractère spécial « = » (ce qui distinguera ce cas particulier lors du décodage)