

EXAMEN (Janvier 2003)

Exercice 1 - (3 pts) -

Ecrire un programme C qui affiche à l'écran les 20 premiers nombres premiers dont la somme des chiffres n'est pas un multiple de 2 (Par exemple 31 n'est pas candidat car $3+1=4$ est un multiple de 2).

Exercice 2 - (8 pts)

Le microprocesseur *Richerium 2003* possède des bus de données et d'adresses de 16 bits. Il est doté de 5 instructions :

push adr	<i>empiler une valeur provenant de la mémoire</i> $SP = SP - 1$ $Pile[SP] = Mem[adr]$
push val	<i>empiler une valeur constante</i> $SP = SP - 1$ $Pile[SP] = val$
calc	<i>calculer</i> $Pile[SP+1] = Pile[SP] - Pile[SP+1]$ $SP = SP + 1$
pop adr	<i>dépiler une valeur et la stocker en mémoire</i> $Mem[adr] = Pile[SP]$ $SP = SP + 1$
jle label	<i>tester le sommet de pile et se déplacer</i> $SP = SP + 1$ if ($Pile[SP-1] \leq 0$) goto label

adr représente une adresse mémoire sur 16 bits, val une constante entière signée et label est une étiquette.

- Soient trois variables a, b, c situées aux adresses mémoire a, b, c . Traduire en langage C les morceaux de Code de la table suivante :

Code 1	Code 2
push a	11: push b
push 0	jle 12
calc	push 1
push c	push b
calc	calc
pop c	pop b
	push 0
	jle 11
	12:

2. On vous demande de coder en *Richerium 2003* les instructions suivantes :

```
c = 0;
if (b < 0) {
    b = -b;
    while (b > 0) {
        c = c + a;
        b = b - 1;
    }
    c = -c;
} else {
if (a < 0) a = -a;
    while (a > b) {
        a = a - b;
        c = c + 1;
    }
}
```

Exercice 3 - (4 pts) - Assembleur

Donner le code assembleur 8086 de la fonction C suivante :

```
int func( int v[], int w[] ) {
    int i, s;
    i = 0; s = 0;
    while ((v[i] != 0) && (w[i] != 0)) {
        s = s + v[i]*w[i]; ++i;
    }
    if (i >= 1) return s/i; else return 0;
}
```

Par convention on considérera que les entiers et les tableaux sont représentés sur 2 octets et que la procédure appelée supprime les paramètres de la pile.

Exercice 4 - (2 pts)

Imaginer un moyen d'échanger deux variables *A* et *B* sans utiliser une troisième variable ou un registre intermédiaire (Penser à XOR).

Exercice 5 - (3 pts)

Répondre aux questions suivantes :

1. Quelles sont les améliorations du Pentium par rapport au 80486?
2. Qu'est ce que Northwood?
3. Quel espace (en nombre d'octets) occupent les nombres à virgule flottante en double précision?
4. Quel autre nom donne t'on également au langage C?
5. Comment peut-on efficacement multiplier un nombre entier par 100 en assembleur sans utiliser la multiplication?

Dans la table précédente R et S représentent l'un des registres AX, BX, CX, DX, SI, DI, BP ou SP avec éventuellement R=S. adr est une valeur sur 16 bits.

Format des instructions <i>mov, add, sub, and, or, xor, cmp</i>	
mov R,[adr] mov R,S mov R,[bx] mov R,[bx+di] mov R,[bx+si] mov R,[bx+di+adr] mov R,[bx+si+adr] mov R,[bp+adr]	mov [adr],R mov [bx],R mov [bx+di],R mov [bx+si],R mov [bx+di+adr],R mov [bx+si+adr],R mov [bp+adr],R
Format des instructions <i>inc, dec, push, pop, not</i>	
inc R	inc [adr]
mul R div R	résultat dans dx:ax = ax × R résultat dans ax, reste dans dx, dx:ax / R
Instructions de saut conditionnel	
je adr (jump on equal) jge adr (jump on greater or equal) jz adr (jump on zero) jmp adr (jump)	jne adr (jum on not equal) jle adr (jump on less or equal) jnz adr (jump on not zero)
call adr (appel de sous-programme)	ret n (retour de sous programme)
shl ax,1 shl ax,cl shr ax,1 shr ax,cl	décalage à gauche décalage à gauche avec cl >1 décalage à droite décalage à droite avec cl >1
loop adr rep stosb rep stosw	décrémente cx et saut à adr si cx <>0 remplir un tableau à l'adresse di sur cx octets avec al remplir un tableau à l'adresse di sur cx*2 octets avec ax

TAB. 1 – Rappel du format des instructions 8086