Organisation des ordinateurs
Examen de première session 2019

Livres fermés. Durée : 3 heures 1/2.

Veuillez répondre à chaque question sur une feuille séparée sur laquelle figurent nom, prénom, matricule et section. Soyez bref et concis, mais précis. Les calculatrices non programmables sont autorisées.

1. En Belgique, un numéro de téléphone mobile est formé
   — d'un préfixe de la forme $04c_1c_2$, où c_1 est un chiffre de 5 à 9 et c_2 un chiffre quelconque, et
   — d'un suffixe composé de 6 chiffres, dont le premier est non nul.

[1/20] (a) En supposant que tous les numéros de téléphone mobile valables sont équiprobables, calculez la quantité d'information contenue dans un numéro.

[1/20] (b) Un opérateur téléphonique gère 2 millions de numéros de téléphone mobile belges, regroupés dans une base de données qui associe à chaque numéro le code postal de son titulaire. En sachant qu'il y a 2825 codes postaux distincts, supposés équiprobables et indépendants des numéros de téléphone, calculez la quantité de mémoire nécessaire au stockage de cette base de données.

[1/20] (c) Quelle quantité d'information un disque dur vendu comme possédant une capacité de 1 TB permet-il de mémoriser ?

[4/20] 2. (a) Quel est le nombre représenté par la constante `0x80000001` dans les représentations
   — par valeur signée ?
   — par complément à un ?
   — par complément à deux ?
   — IEEE754 simple précision ?

[1/20] (b) Quels sont les nombres dont les représentations sur n bits par complément à un et par complément à deux sont égales ? (Justifiez votre réponse.)

[1/20] (c) En utilisant la notation hexadécimale, donnez une représentation d'une valeur indéterminée (*Not A Number*) dans le système IEEE754 en double précision.

[1/20] 3. (a) À quoi servent les drapeaux d'un processeur ? Comment un programmeur peut-il les utiliser ?

(b) Décrivez, le plus simplement possible, l'effet des fragments de code assembleur x86-64 suivants. (On suppose que la pile est correctement configurée avant leur exécution.)

[2/20] — Fragment 1 :

```
PUSH RBX
PUSH RAX
MOV  AL, byte ptr[RSP+0]
MOV  BL, byte ptr[RSP+7]
MOV  byte ptr [RSP+0], BL
MOV  byte ptr [RSP+7], AL
POP  RAX
POP  RBX
```

[2/20] — Fragment 2 :

```
   PUSH RDX
   CMP  AX, 0
   JGE  p
n: MOV RDX, -1
   MOV DX, AX
   JMP f
p: MOV RDX, 0
   MOV DX, AX
f: MOV RAX, RDX
   POP RDX
```

[1/20] (c) Expliquez comment une valeur est représentée en mémoire dans le système petit-boutiste.

4. Dans le cadre du développement d'une application de calcul statistique, on souhaite programmer une fonction `histogramme` capable de calculer l'histogramme d'un tableau d'octets donné. Cette fonction prend pour entrée

— l'adresse d'un tableau d'octets `ts`,
— la taille `n` de ce tableau, exprimée sur 64 bits,
— l'adresse d'un tableau `th` contenant 256 entiers de 64 bits, non initialisés.

À l'issue de l'exécution de cette fonction, chaque case de `th` doit contenir le nombre d'octets de `ts` égaux à son indice. Par exemple, la cinquième case `th[4]` de `th` contiendra le nombre d'octets de `ts` égaux à 4. La fonction ne retourne rien.

[1/20] — Écrire, en pseudocode ou en langage C (au choix), un algorithme permettant de résoudre ce problème.

[4/20] — Traduire cet algorithme en assembleur x86-64, en veillant à respecter la convention d'appel de fonctions des systèmes *Unix*.

Annexe

Code ASCII

20		30	0	40	@	50	P	60	‘	70	p
21	!	31	1	41	A	51	Q	61	a	71	q
22	”	32	2	42	B	52	R	62	b	72	r
23	#	33	3	43	C	53	S	63	c	73	s
24	$	34	4	44	D	54	T	64	d	74	t
25	%	35	5	45	E	55	U	65	e	75	u
26	&	36	6	46	F	56	V	66	f	76	v
27	’	37	7	47	G	57	W	67	g	77	w
28	(	38	8	48	H	58	X	68	h	78	x
29	)	39	9	49	I	59	Y	69	i	79	y
2A	*	3A	:	4A	J	5A	Z	6A	j	7A	z
2B	+	3B	;	4B	K	5B	[	6B	k	7B	{
2C	,	3C	<	4C	L	5C	\	6C	l	7C	|
2D	-	3D	=	4D	M	5D	]	6D	m	7D	}
2E	.	3E	>	4E	N	5E	^	6E	n	7E	~
2F	/	3F	?	4F	O	5F	_	6F	o		

UTF-8

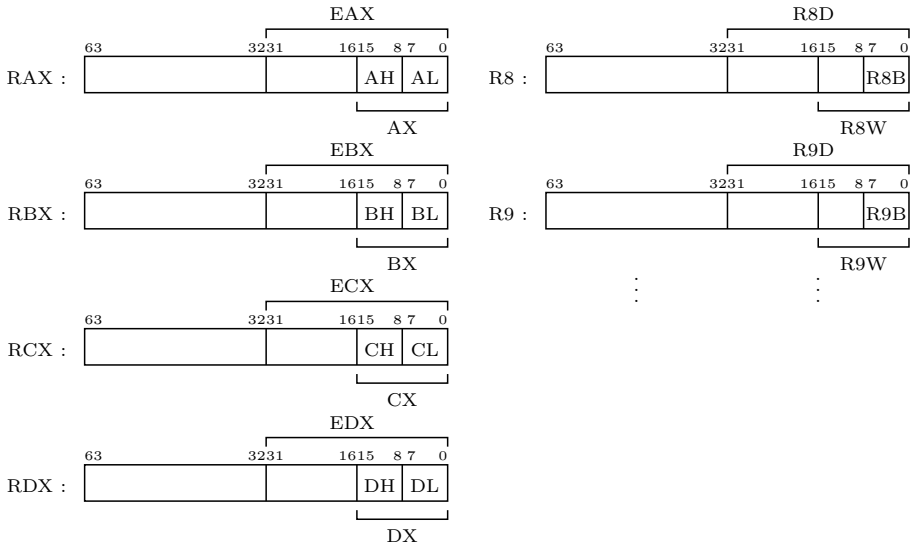
— [0, 0x7F] : $\boxed{0b_6b_5\dots b_0}$

— [0x80, 0x7FF] : $\boxed{110b_{10}b_9\dots b_6}$ $\boxed{10b_5b_4\dots b_0}$

— [0x800, 0xFFFF] : $\boxed{1110b_{15}b_{14}b_{13}b_{12}}$ $\boxed{10b_{11}b_{10}\dots b_6}$ $\boxed{10b_5b_4\dots b_0}$

— [0x10000, 0x10FFFF] : $\boxed{11110b_{20}b_{19}b_{18}}$ $\boxed{10b_{17}b_{16}\dots b_{12}}$ $\boxed{10b_{11}b_{10}\dots b_6}$ $\boxed{10b_5b_4\dots b_0}$

Registres x86-64



Modes d'adressage des instructions x86-64

`MOV, ADD, SUB, CMP, AND, OR, XOR`	
Op. 1	Op. 2
reg	*imm*
mem	*imm*
reg	*reg*
reg	*mem*
mem	*reg*

`XCHG`	
Op. 1	Op. 2
reg	*reg*
reg	*mem*
mem	*reg*

`INC, DEC, NOT, POP`
Op. 1
reg
mem

`MUL, IMUL, PUSH, JMP, Jxx, LOOP, CALL`
Op. 1
imm
reg
mem

Drapeaux affectés par les instructions x86-64

	CF	ZF	SF	OF
`MOV, XCHG, NOT, PUSH, POP, JMP, Jxx, LOOP, CALL, RET`	–	–	–	–
`ADD, SUB, CMP`	✓	✓	✓	✓
`AND, OR, XOR`	0	✓	✓	0
`INC, DEC`	–	✓	✓	✓
`MUL, IMUL`	✓	?	?	✓

Instructions de saut conditionnel x86-64

Instruction	Condition
`JC`	CF = 1
`JNC`	CF = 0
`JZ`	ZF = 1
`JNZ`	ZF = 0
`JS`	SF = 1
`JNS`	SF = 0
`JO`	OF = 1
`JNO`	OF = 0

Instruction	Condition	
`JE`	$op1 = op2$	
`JNE`	$op1 \neq op2$	
`JG`	$op1 > op2$	(valeurs signées)
`JGE`	$op1 \geq op2$	(valeurs signées)
`JL`	$op1 < op2$	(valeurs signées)
`JLE`	$op1 \leq op2$	(valeurs signées)
`JA`	$op1 > op2$	(valeurs non signées)
`JAE`	$op1 \geq op2$	(valeurs non signées)
`JB`	$op1 < op2$	(valeurs non signées)
`JBE`	$op1 \leq op2$	(valeurs non signées)

Convention d'appel de fonctions Unix

— Six premiers arguments : Registres RDI, RSI, RDX, RCX, R8 et R9.
— Valeur de retour : Registre RAX.
— Registres à préserver : RBX, RBP, R12, R13, R14 et R15.