Organisation des ordinateurs
Examen de première session 2018

Livres fermés. Durée : 3 heures 1/2.

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

[2/20] 1. (a) En français, la probabilité qu'une lettre prise au hasard dans un texte soit un "O", un "C", un "T" ou un "E" est respectivement égale, approximativement, à 5,02%, 3,18%, 5,92% et 12,20%. Par souci de simplicité, on considère que ces probabilités ne dépendent ni de la place des lettres dans les mots, ni de la nature des lettres voisines.

Sous ces hypothèses, on demande de calculer la quantité d'information contenue dans le mot "OCTET".

[1/20] (b) Dans les ordinateurs, pourquoi représente-t-on l'information à l'aide de signaux discrets plutôt que continus ?

[4/20] 2. (a) Quels sont les plus petits et les plus grands nombres représentables à l'aide des encodages

i. entier non signé sur 16 bits ?

ii. entier par complément à deux sur 16 bits ?

iii. en virgule fixe par complément à deux, avec 8 bits avant et 8 bits après la virgule ?

iv. par le procédé IEEE 754 en double précision ?

[1/20] (b) Calculer le produit $-1 \times (-1)$ à l'aide de la représentation par complément à deux des entiers sur 4 bits.

[1/20] (c) Quel est le nombre représenté par la suite de bits

$$\boxed{11000000000000000000000000000000}$$

(il y a 2 bits égaux à 1 suivis de 30 bits égaux à 0), par le procédé IEEE 754 ?

[1/20] 3. (a) A quoi sert la mémoire de masse d'un ordinateur ? Quelles sont les différences entre ce type de mémoire et la mémoire vive ?

[2/20] (b) Expliquer le principe et les modalités d'utilisation de l'adressage indirect indexé de l'architecture x86-64.

[2/20] (c) Le plus simplement possible, décrivez l'effet des instructions x86-64 suivantes :

i. `AND word ptr[R8], 0xff`

ii. `CMP EAX, EAX`

iii. `PUSH qword ptr[RAX]`

iv. `RET`

4. On souhaite programmer une fonction `prefixe_commun(t1, t2, n)` chargée de déterminer la longueur du préfixe commun entre les deux tableaux d'octets pointés par `t1` et `t2`, tous deux de taille `n` $\in [0, 2^{32} - 1]$. En d'autres termes, cette fonction doit déterminer la plus grande valeur k telle que `t1`$[0] =$ `t2`$[0]$, `t1`$[1] =$ `t2`$[1]$, ..., `t1`$[k-1] =$ `t2`$[k-1]$, et retourner cette valeur. Dans le cas où `t1`$[0] \neq$ `t2`$[0]$, la fonction doit retourner 0.

[2/20] (a) Écrire, en pseudocode ou en langage C (au choix), un algorithme permettant de résoudre ce problème.

[4/20] (b) 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

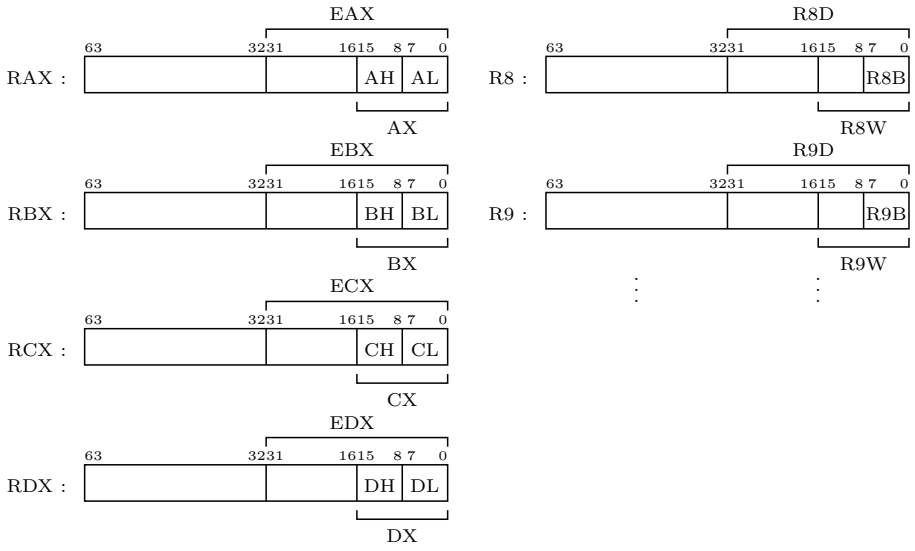
— [0, 0x7F] : $\boxed{0b_6b_5\dots b_0}$

— [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.