Organisation des ordinateurs
Examen de seconde 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 développements doivent être justifiés. Les calculatrices non programmables sont autorisées.

[1/20] 1. (a) Quelle est la quantité d'information fournie par k signaux pouvant chacun prendre n valeurs équiprobables ?

[2/20] (b) Une bande magnétique mesurant 154 m est composée de 1200 pistes parallèles. Sur chacune de ces pistes, l'information est représentée par l'orientation de domaines magnétiques disposés séquentiellement. Chaque domaine mesure 6,8 μm de longueur et possède quatre orientations possibles. On demande de calculer la quantité d'information totale mémorisée par une telle bande magnétique.

[3/20] 2. (a) Quelle est la représentation sur n bits (avec $n \geq 2$) du nombre -1 par les procédés

i. par complément à un ?

ii. par complément à deux ?

iii. en virgule fixe par complément à deux avec 1 bit après la virgule ?

[1/20] (b) Dans le procédé de représentation IEEE 754 en simple précision, quel est le plus petit nombre strictement positif représentable ?

[1/20] (c) Calculer la somme $-2+(-1)$ à l'aide de la représentation par complément à un des entiers sur 4 bits.

[1/20] (d) Quels sont les avantages du procédé par complément à deux pour représenter les nombres entiers signés ?

[1/20] 3. (a) Qu'est-ce que le code machine ? En quoi diffère-t-il du code assembleur ?

[2/20] (b) Dans un programme assembleur x86-64, on souhaite écrire la valeur v dans la k-ème case d'un tableau d'entiers (de 32 bits) pointé par le registre RAX. Le premier élément du tableau correspond à $k = 1$, le second à $k = 2$, et ainsi de suite. Les valeurs de v et de k sont respectivement disponibles dans les registres R8D et RSI. On demande d'écrire une instruction x86-64 réalisant cette opération d'écriture.

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

i. `IMUL AX`

ii. `SUB R8D, -1`

iii. `POP qword ptr[0x100]`

iv. `CALL qword ptr[8 * RAX]`

4. On souhaite programmer une fonction `facteur_deux(n)` chargée de calculer le plus grand entier k tel que 2^k divise n, où n est un nombre de 64 bits supposé strictement positif. Par exemple, `facteur_deux(96)` doit retourner 5, car $96 = 2^5\,3$. Lorsque `n` est impair, `facteur_deux(n)` doit retourner 0.

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

Suggestion : En exploitant les instructions logiques, compter le nombre de bits nuls situés à la fin de la représentation binaire de n.

[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

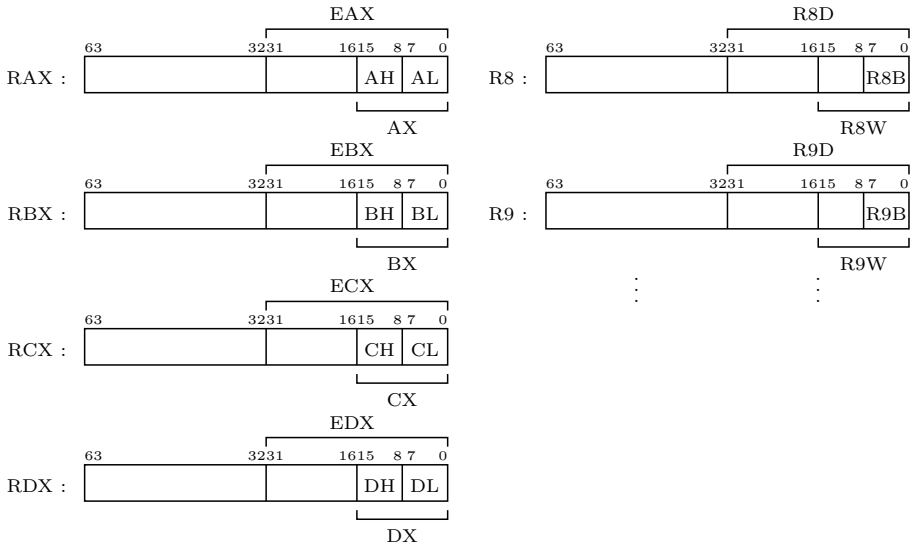
— [0, 0x7F] : $\boxed{0b_6b_5\dots b_0}$

— [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.