Organisation des ordinateurs
Examen d'août 2025

Livres fermés. Durée : 3 heures 30

Veuillez répondre aux questions sur des feuilles séparées sur lesquelles figurent votre nom, votre prénom et votre matricule. Les calculatrices non programmables sont autorisées.

[2/20] 1. (a) On effectue une expérience qui consiste à lancer trois fois de suite une pièce de monnaie non truquée, et à compter le nombre de fois parmi ces trois lancers qu'elle retombe sur son côté pile. Quelle quantité d'information obtient-on dans le cas où le résultat de cette expérience vaut 2 ?

[1/20] (b) Quelle quantité d'information, exprimée en térabits, un disque dur vendu comme ayant une capacité de 12 téraoctets permet-il de mémoriser ?

[1/20] 2. (a) Donner la représentation hexadécimale sur 16 bits du nombre -2025 selon les encodages
— par valeur signée,
— par complément à un, et
— par complément à deux
des entiers.

[2/20] (b) Calculer le plus précisément possible la somme $-1 + \frac{7}{16}$ en complément à deux en virgule fixe avec 3 bits avant et 3 bits après la virgule.

[1/20] (c) Calculer le produit $-4 \,.\, (-4)$ à partir de la représentation en complément à deux sur n bits des nombres, où n est le plus petit entier qui permet d'effectuer l'opération sans dépassement arithmétique.

[2/20] (d) Donner la représentation en simple précision du nombre $1{,}5 \,.\, 2^{-130}$ selon le standard IEEE754. Cette représentation est-elle exacte ou approximée ? (Justifier votre réponse.)

[1/20] 3. (a) En quoi la mémoire vive et la mémoire morte diffèrent-elles ? Donner un exemple d'utilisation de chacun de ces types de mémoire dans un ordinateur moderne.

[2/20] (b) Décrire trois modes d'adressage de votre choix définis par l'architecture x86-64. Illustrer chacun d'entre eux à l'aide d'un exemple concret d'instruction qui l'utilise, en précisant la valeur des registres concernés avant et après l'exécution de l'instruction.

[3/20] (c) Expliquer étape par étape comment se déroulera l'exécution du fragment de code assembleur x86-64 suivant, en faisant l'hypothèse que la pile est initialement correctement configurée. Quelle sera la valeur finale de `RAX` ?

```
MOV  RAX, 0x1908
MOV  RBX, 0x2025
PUSH RAX
PUSH RBX
POP  RAX
ADD  RAX, qword ptr[RSP]
POP  RBX
SUB  RSP, 8
XOR  RBX, RAX
MOV  qword ptr[RSP], RBX
POP  RAX
```

4. On souhaite programmer une fonction `nb_changements_signe` acceptant en arguments un tableau d'entiers signés `t` représentés sur 16 bits, et le nombre d'éléments `n` de ce tableau donné sous la forme d'un entier non signé de 32 bits. Cette fonction doit retourner le nombre de changements de signe entre les paires d'éléments consécutifs du tableau. On considère qu'il y a un changement de signe chaque fois que deux éléments consécutifs $\texttt{t}[i]$ et $\texttt{t}[i+1]$ sont tels qu'un des deux est positif ou nul et l'autre strictement négatif. Par exemple, si `t` contient $[-2, 1, -1, 3]$, alors la fonction doit retourner 3. Si `t` contient $[1, 0, 1]$ ou si `t` est vide, alors la fonction doit retourner 0.

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

[4/20] (b) Traduire cet algorithme en un programme 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

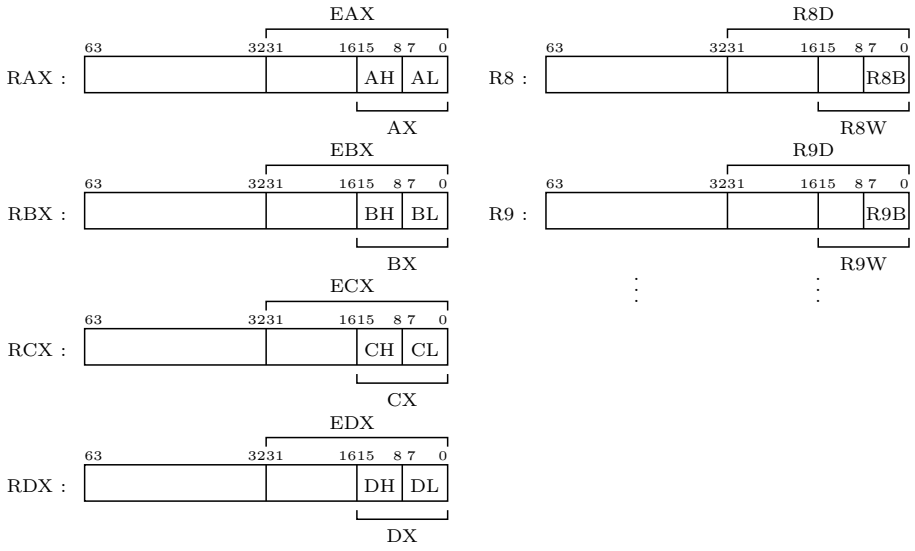
— [0, 0x7F] : $\boxed{0b_6b_5\dots b_0}$

— [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, MUL, IMUL`
Op. 1
reg
mem

`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.