Organisation des ordinateurs
Examen de seconde session 2024

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.

1. Une machine analyse des échantillons de roche pour détecter la présence d'un type spécifique de minéral, qui a une probabilité de 15% d'être présent. Toutes les 5 millisecondes, la machine envoie un signal à un ordinateur pour indiquer la présence ou l'absence de ce minéral.

[2/20] (a) Calculer la quantité d'information moyenne contenue dans un signal émis par la machine.

[1/20] (b) Quelle quantité moyenne de mémoire vive l'ordinateur doit-il allouer pour retenir les données transmises par la machine pendant 10 heures de mesures ininterrompues ?

[1/20] 2. (a) Quel sont les nombres entiers possédant la représentation hexadécimale 0xC49, selon les représentations
— signée
— par complément à un
— par complément à deux
sur 12 bits ?

[1/20] (b) Calculer la somme $(-\frac{5}{2}) + \frac{21}{8}$ en complément à deux en virgule fixe avec 3 bits avant et 3 bits après la virgule.

[2/20] (c) Calculer la somme $30 + (-18)$ en complément à un sur 6 bits.

[2/20] (d) En plus des représentations en simple et en double précision vues au cours, la norme IEEE754 définit également une représentation en demi-précision. Celle-ci est basée sur les mêmes principes, mais représente les nombres réels sur 16 bits, décomposés en 1 bit de signe, 5 bits pour l'exposant et 10 bits pour la mantisse.

Quels sont le plus petit nombre positif différent de 0 et le plus grand nombre représentables de façon exacte en demi-précision ? On demande de donner ces nombres, ainsi que leur représentation dans ce format.

[1/20] 3. (a) Qu'est-ce que la mémoire morte, et à quoi sert-elle ? Donner un exemple de technologie de mémoire morte et en expliquer les spécificités.

[1/20] (b) En quoi consiste l'adressage indirect indexé ? Donner un exemple d'instruction assembleur utilisant ce type d'adressage, et expliquer l'opération qu'elle effectue.

[3/20] (c) Expliquer étape par étape comment se déroulera l'exécution du fragment de code assembleur x86-64 suivant. Quelle sera la valeur finale de `RAX` ?

```
MOV RAX, 0x101
MOV dword ptr[0x100], EAX
ADD word ptr[RAX - 1], AX
ADD AL, byte ptr[RAX]
XOR byte ptr[RAX - 1], AH
OR byte ptr[RAX], AH
MOV AX, word ptr[RAX - 1]
ADD byte ptr[RAX], 0x11
MOV AH, byte ptr[RAX]
```

4. On souhaite programmer une fonction `f` acceptant en arguments un pointeur vers un tableau d'octets `t` et la taille `n` de ce tableau. On suppose que `t` contient une séquence de caractères Unicode représentés correctement selon le procédé d'encodage UTF-8. La fonction `f` doit retourner le nombre de caractères contenus dans `t`.

   Par exemple, si `t` contient les 6 octets 0x31, 0x30, 0x20, 0xE2, 0x82 et 0xAC, représentant la séquence de caractères "10 €", alors la fonction `f` doit retourner 4.

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

[5/20] (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

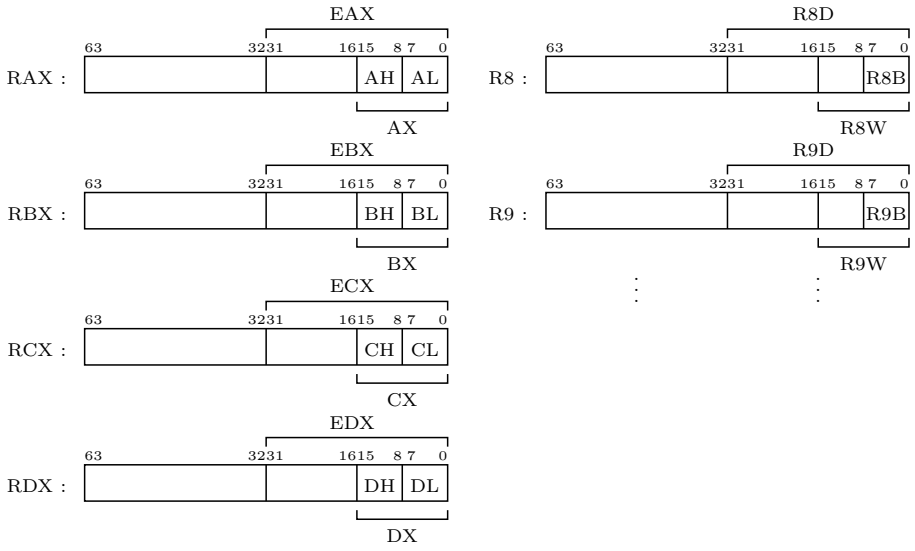
— [0, 0x7F] : $\boxed{0b_6b_5\dots b_0}$

— [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.