Pas de QCM 2 en algo

Graphes
QCM ~~3~~ 2
10 novembre 2025

1. **Soit G=<S,A> un graphe orienté d'ordre N avec M=Card(A), si G est représenté par listes d'adjacence, la complexité du parcours profondeur de G est**
   (a) $O(N)$
   (b) $O(max(N, M))$
   (c) $O(N^2)$
   (d) $O(N.M)$
   (e) $O(M^2)$

2. **Dans le parcours profondeur d'un graphe non orienté, $x \to y$ est un arc en arrière si dans la forêt couvrante**
   (a) y est le père de x
   (b) x est un descendant de y
   (c) x est un ascendant de y
   (d) x n'est ni un descendant ni un ascendant de y

3. **Si dans un graphe orienté il existe un chemin $x \rightsquigarrow y$ alors dans la forêt couvrante du parcours profondeur lancé depuis le sommet y**
   (a) x est y peuvent être dans des arbres différents
   (b) x et y sont toujours dans le même arbre
   (c) x et y ne sont jamais dans le même arbre

4. **Supposons que $pref[i]$ est le numéro d'ordre préfixe de rencontre d'un sommet i lors du parcours en profondeur d'un graphe orienté.**
   Les arcs $x \to y$ tels que $pref[x] < pref[y]$ peuvent être
   (a) des arcs couvrants
   (b) des arcs en arrière
   (c) des arcs en avant
   (d) des arcs croisés

5. **Dans un parcours en profondeur d'un graphe orienté G, les arcs $x \to y$ tels qu'il n'existe pas de chemin $x \rightsquigarrow y$ dans la forêt couvrante sont**
   (a) des arcs couvrants
   (b) des arcs en arrière
   (c) des arcs en avant
   (d) des arcs croisés

Soit le graphe non orienté G, dans lequel les sommets sont numérotés de 1 à 7, représenté par la matrice adjacence suivante dans laquelle 0 = faux et 1 = vrai.

	1	2	3	4	5	6	7
1					1		1
2			1			1	
3		1				1	
4					1		1
5	1			1			1
6		1	1				
7	1			1	1		

Toutes les questions suivantes portent sur le parcours profondeur de G à partir du sommet 1 en choisissant les sommets en ordre croissant et sur la forêt couvrante (à laquelle on ajoute les autres arcs) associée.

6. **Combien d'arbres contient la forêt couvrante ?**
   (a) 1
   (b) 2
   (c) 3
   (d) 4

7. **Quel est l'ordre préfixe de rencontre des sommets ?**
   (a) 1, 2, 3, 4, 5, 6, 7
   (b) 1, 5, 4, 7, 2, 3, 6
   (c) 1, 5, 7, 4, 2, 3, 6
   (d) 7, 4, 5, 1, 6, 3, 2
   (e) 4, 5, 7, 1, 3, 6, 2

8. **Quel est l'ordre suffixe de rencontre des sommets ?**
   (a) 1, 2, 3, 4, 5, 6, 7
   (b) 1, 5, 4, 7, 2, 3, 6
   (c) 1, 5, 7, 4, 2, 3, 6
   (d) 7, 4, 5, 1, 6, 3, 2
   (e) 4, 5, 7, 1, 3, 6, 2

9. **Combien d'arcs en arrière y a-t-il dans la forêt couvrante ?**
   (a) 0
   (b) 1
   (c) 2
   (d) 3
   (e) 4

10. **Dans la forêt couvrante, l'arc $1 \to 7$**
    (a) est un arc couvrant
    (b) est un arc en arrière
    (c) est un arc en avant
    (d) est un arc croisé
    (e) n'existe pas

QCM N°5

Lundi 10 novembre 2025

Question 11

Dans $E = \mathbb{R}^3$, on considère la famille $\mathcal{F} = \big(u_1{=}(1,0,0), u_2{=}(1,1,0), u_3{=}(4,1,0)\big)$. Alors :

a. $\mathcal{F}$ est libre dans E

b. $\mathcal{F}$ est une famille génératrice de E

c. Aucun des autres choix

Question 12

Dans $E = \mathbb{R}^3$, on considère la famille $\mathcal{F} = \big(u_1{=}(1,0,0), u_2{=}(1,1,0), u_3{=}(4,1,1)\big)$. Alors :

a. $\mathcal{F}$ est libre dans E

b. $\mathcal{F}$ est une famille génératrice de E

c. Aucun des autres choix

Question 13

Dans $E = \mathbb{R}^3$, on considère la famille $\mathcal{F} = \big(u_1{=}(1,0,0), u_2{=}(1,1,0), u_3{=}(4,1,0)\big)$. Alors :

a. $\text{Vect}\,\mathcal{F} = \{0_E\}$

b. $\text{Vect}\,\mathcal{F}$ est une droite

c. $\text{Vect}\,\mathcal{F}$ est un plan

d. $\text{Vect}\,\mathcal{F} = E$

e. Aucun des autres choix

Question 14

Dans $E = \mathbb{R}^3$, on considère la famille $\mathcal{F} = \big(u_1{=}(1,1,-1), u_2{=}(-1,-1,1), u_3{=}(2,2,-2)\big)$. Alors :

a. $\text{Vect}\,\mathcal{F} = \{0_E\}$

b. $\text{Vect}\,\mathcal{F}$ est une droite

c. $\text{Vect}\,\mathcal{F}$ est un plan

d. $\text{Vect}\,\mathcal{F} = E$

e. Aucun des autres choix

Question 15

Dans $E = \mathbb{R}_2[X]$, on considère la famille $\mathcal{F} = (X-1, X^2-X, X^2-1)$. Alors la dimension de $\text{Vect}\,\mathcal{F}$ vaut :

a. $\dim(\text{Vect}\,\mathcal{F}) = 0$

b. $\dim(\text{Vect}\,\mathcal{F}) = 1$

X c. $\dim(\text{Vect}\,\mathcal{F}) = 2$

X d. $\dim(\text{Vect}\,\mathcal{F}) = 3$

e. Aucun des autres choix

Question 16

Dans $E = \mathbb{R}[X]$, on considère une famille de la forme $\mathcal{F} = (P_1, P_2, P_3, P_4)$. Alors on sait que :

a. La famille $\mathcal{F}$ est libre

X b. La famille $\mathcal{F}$ est liée

c. $\mathcal{F}$ est une famille génératrice de E

X d. $\mathcal{F}$ n'est pas une famille génératrice de E

e. Aucun des autres choix

Question 17

Dans $E = \mathbb{R}_2[X]$, on considère une famille de la forme $\mathcal{F} = (P_1, P_2)$. Alors on sait que :

a. La famille $\mathcal{F}$ est libre

b. La famille $\mathcal{F}$ est liée

c. $\mathcal{F}$ est une famille génératrice de E

X X d. $\mathcal{F}$ n'est pas une famille génératrice de E

e. Aucun des autres choix

Question 18

Dans $E = \mathbb{R}^2$, on considère la base $\mathcal{B} = (\varepsilon_1{=}(1,-1), \varepsilon_2 = (2,1))$.

Soit $u \in E$ dont les coordonnées dans $\mathcal{B}$ sont $X = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$. Alors ce vecteur u est égal à :

a. $u = \left(-\frac{1}{3}, \frac{2}{3}\right)$

b. $u = \left(\frac{1}{3}, -\frac{2}{3}\right)$

✗ c. $u = (3,0)$

d. $u = (1,1)$

e. Aucun des autres choix

Question 19

Dans $E = \mathbb{R}^2$, on considère la base $\mathcal{B} = (\varepsilon_1{=}(1,-1), \varepsilon_2 = (2.1))$.

Soit $u = (3,4) \in E$. Pour déterminer les coordonnées $X = \begin{pmatrix} x \\ y \end{pmatrix}$ du vecteur u dans la base $\mathcal{B}$:

✗ a. On résout le système $\begin{cases} x + 2y &= 3 \\ -x + y &= 4 \end{cases}$

b. On résout le système $\begin{cases} x - y &= 3 \\ 2x + y &= 4 \end{cases}$

c. On pose : $\begin{cases} 3 + 2{\times}4 &= x \\ -3 + 4 &= y \end{cases}$

d. On pose : $\begin{cases} 3 - 4 &= x \\ 2{\times}3 + 4 &= y \end{cases}$

e. Aucun des autres choix

Question 20

Dans $E = \mathbb{R}_2[X]$, on considère la famille $\mathcal{F} = (X - 1, X^2 - 3)$. Alors :

✗ a. On peut obtenir une base de E en ajoutant des éléments à $\mathcal{F}$

b. On peut obtenir une base de E en retirant des éléments à $\mathcal{F}$

c. Aucun des autres choix

QCM 3
Architecture des ordinateurs

Lundi 10 novembre 2025

Pour toutes les questions, une ou plusieurs réponses sont possibles.

21. Le 68000 possède :
    - A. Aucune de ces réponses.
    - B. Un pointeur de pile : PC
    - C. Deux pointeurs de pile : USP et SSP
    - D. Deux pointeurs de pile : CCR et SR

22. À quoi sert le symbole '#' ?
    - A. Il indique qu'un opérande est une adresse.
    - B. Il indique qu'un opérande est sous forme hexadécimale.
    - C. Il indique qu'un opérande est sous forme décimale.
    - D. Il indique qu'un opérande est une donnée immédiate.

23. Soit l'instruction suivante : `MOVE.L -4(A0),D0`
    - A. A0 est décrémenté de 4.
    - B. A0 ne change pas.
    - C. A0 est décrémenté de 2.
    - D. A0 est décrémenté de 1.

24. Quelle instruction sert essentiellement à appeler un sous-programme ?
    - A. JMP
    - B. RTS
    - C. JSR
    - D. Aucune de ces réponses.

25. Après l'exécution d'une instruction `RTS`, le pointeur de pile est :
    - A. Incrémenté de quatre.
    - B. Incrémenté de deux.
    - C. Décrémenté de deux.
    - D. Aucune de ces réponses.

26. Les étapes pour dépiler une donnée sont :
    - A. Écrire dans la mémoire pointée par A7 puis décrémenter A7.
    - B. Décrémenter A7 puis écrire dans la mémoire pointée par A7.
    - C. Lire la mémoire pointée par A7 puis incrémenter A7.
    - D. Décrémenter A7 puis lire la mémoire pointée par A7.

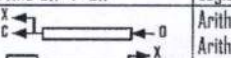
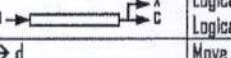
27. Choisir les réponses correctes.
   - [x] A. Un nouvel élément est toujours ajouté au sommet de la pile.
   - [x] B. Un élément est toujours retiré du sommet de la pile.
   - [ ] C. Un nouvel élément est toujours ajouté au bas de la pile.
   - [ ] D. Un élément est toujours retiré du bas de la pile.

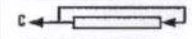
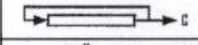
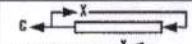
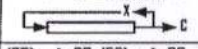
28. Pour empiler une donnée :
   - [x] A. On décrémente A7 d'abord.
   - [ ] B. On incrémente A7 d'abord.
   - [ ] C. On ne change pas A7.
   - [ ] D. Aucune de ces réponses.

29. Soit l'instruction suivante : `MOVEM.L D1-D3/A4/A5,-(A7)`
    Quelle instruction est équivalente ?
   - [ ] A. `MOVEM.L D1/D3/A4-A5,-(A7)`
   - [ ] B. `MOVEM.L D1/D3/A4/A5,-(A7)`
   - [x] C. `MOVEM.L A4/A5/D1/D2/D3,-(A7)`
   - [ ] D. Aucune de ces réponses.

30. Le registre A7 :
   - [ ] A. Pointe le bas de la pile.
   - [ ] B. Pointe le milieu de la pile.
   - [x] C. Pointe le sommet de la pile.
   - [ ] D. Aucune de ces réponses.

EASy68K Quick Reference v1.8 http://www.wowgwep.com/EASy68K.htm

Opcode	Size	Operand	CCR	Effective Address s=source, d=destination, e=either, i=displacement												Operation	Description
	BWL	s,d	XNZVC	Dn	An	(An)	(An)+	-(An)	(i,An)	(i,An,Rn)	abs.W	abs.L	(i,PC)	(i,PC,Rn)	#n		
ABCD	B	Dy,Dx	*U*U*	e	-	-	-	-	-	-	-	-	-	-	-	$Dy_{(0)} + Dx_{(0)} + X \rightarrow Dx_{(0)}$	Add BCD source and eXtend bit to
		-(Ay),-(Ax)		-	-	-	-	e	-	-	-	-	-	-	-	$-(Ay)_{(0)} + -(Ax)_{(0)} + X \rightarrow -(Ax)_{(0)}$	destination, BCD result
ADD [4]	BWL	s,Dn	*****	e	s	s	s	s	s	s	s	s	s	s	s[4]	s + Dn → Dn	Add binary (ADDI or ADDQ is used when
		Dn,d		e	d[4]	d	d	d	d	d	d	d	-	-	-	Dn + d → d	source is #n. Prevent ADDQ with #n.L)
ADDA [4]	WL	s,An	-----	s	e	s	s	s	s	s	s	s	s	s	s	s + An → An	Add address (.W sign-extended to .L)
ADDI [4]	BWL	#n,d	*****	d	-	d	d	d	d	d	d	d	-	-	s	#n + d → d	Add immediate to destination
ADDQ [4]	BWL	#n,d	*****	d	d	d	d	d	d	d	d	d	-	-	s	#n + d → d	Add quick immediate (#n range: 1 to 8)
ADDX	BWL	Dy,Dx	*****	e	-	-	-	-	-	-	-	-	-	-	-	Dy + Dx + X → Dx	Add source and eXtend bit to destination
		-(Ay),-(Ax)		-	-	-	-	e	-	-	-	-	-	-	-	-(Ay) + -(Ax) + X → -(Ax)	
AND [4]	BWL	s,Dn	-**00	e	-	s	s	s	s	s	s	s	s	s	s[4]	s AND Dn → Dn	Logical AND source to destination
		Dn,d		e	-	d	d	d	d	d	d	d	-	-	-	Dn AND d → d	(ANDI is used when source is #n)
ANDI [4]	BWL	#n,d	-**00	d	-	d	d	d	d	d	d	d	-	-	s	#n AND d → d	Logical AND immediate to destination
ANDI [4]	B	#n,CCR	=====	-	-	-	-	-	-	-	-	-	-	-	s	#n AND CCR → CCR	Logical AND immediate to CCR
ANDI [4]	W	#n,SR	=====	-	-	-	-	-	-	-	-	-	-	-	s	#n AND SR → SR	Logical AND immediate to SR (Privileged)
ASL	BWL	Dx,Dy	*****	e	-	-	-	-	-	-	-	-	-	-	-		Arithmetic shift Dy by Dx bits left/right
ASR		#n,Dy		d	-	-	-	-	-	-	-	-	-	-	s		Arithmetic shift Dy #n bits L/R (#n: 1 to 8)
	W	d		-	-	d	d	d	d	d	d	d	-	-	-		Arithmetic shift ds 1 bit left/right (.W only)
Bcc	BW[3]	address[2]	-----	-	-	-	-	-	-	-	-	-	-	-	-	if cc true then	Branch conditionally (cc table on back)
																address → PC	(8 or 16-bit ± offset to address)
BCHG	B L	Dn,d	--*--	e[1]	-	d	d	d	d	d	d	d	-	-	-	NOT(bit number of d) → Z	Set Z with state of specified bit in d then
		#n,d		d[1]	-	d	d	d	d	d	d	d	-	-	s	NOT(bit n of d) → bit n of d	invert the bit in d
BCLR	B L	Dn,d	--*--	e[1]	-	d	d	d	d	d	d	d	-	-	-	NOT(bit number of d) → Z	Set Z with state of specified bit in d then
		#n,d		d[1]	-	d	d	d	d	d	d	d	-	-	s	0 → bit number of d	clear the bit in d
BRA	BW[3]	address[2]	-----	-	-	-	-	-	-	-	-	-	-	-	-	address → PC	Branch always (8 or 16-bit ± offset to addr)
BSET	B L	Dn,d	--*--	e[1]	-	d	d	d	d	d	d	d	-	-	-	NOT(bit n of d) → Z	Set Z with state of specified bit in d then
		#n,d		d[1]	-	d	d	d	d	d	d	d	-	-	s	1 → bit n of d	set the bit in d
BSR	BW[3]	address[2]	-----	-	-	-	-	-	-	-	-	-	-	-	-	PC → -(SP); address → PC	Branch to subroutine (8 or 16-bit ± offset)
BTST	B L	Dn,d	--*--	e[1]	-	d	d	d	d	d	d	d	d	d	-	NOT(bit Dn of d) → Z	Set Z with state of specified bit in d
		#n,d		d[1]	-	d	d	d	d	d	d	d	d	d	s	NOT(bit #n of d) → Z	Leave the bit in d unchanged
CHK	W	s,Dn	-*UUU	e	-	s	s	s	s	s	s	s	s	s	s	if Dn<0 or Dn>s then TRAP	Compare Dn with 0 and upper bound [s]
CLR	BWL	d	-0100	d	-	d	d	d	d	d	d	d	-	-	-	0 → d	Clear destination to zero
CMP [4]	BWL	s,Dn	-****	e	s[4]	s	s	s	s	s	s	s	s	s	s[4]	set CCR with Dn - s	Compare Dn to source
CMPA [4]	WL	s,An	-****	s	e	s	s	s	s	s	s	s	s	s	s	set CCR with An - s	Compare An to source
CMPI [4]	BWL	#n,d	-****	d	-	d	d	d	d	d	d	d	-	-	s	set CCR with d - #n	Compare destination to #n
CMPM [4]	BWL	(Ay)+,(Ax)+	-****	-	-	-	e	-	-	-	-	-	-	-	-	set CCR with (Ax) - (Ay)	Compare (Ax) to (Ay); Increment Ax and Ay
DBcc	W	Dn,addres[2]	-----	-	-	-	-	-	-	-	-	-	-	-	-	if cc false then { Dn-1 → Dn	Test condition, decrement and branch
																if Dn <> -1 then addr →PC }	(16-bit ± offset to address)
DIVS	W	s,Dn	-***0	e	-	s	s	s	s	s	s	s	s	s	s	±32bit Dn / ±16bit s → ±Dn	Dn= [16-bit remainder, 16-bit quotient]
DIVU	W	s,Dn	-***0	e	-	s	s	s	s	s	s	s	s	s	s	32bit Dn / 16bit s → Dn	Dn= [16-bit remainder, 16-bit quotient]
EOR [4]	BWL	Dn,d	-**00	e	-	d	d	d	d	d	d	d	-	-	s[4]	Dn XOR d → d	Logical exclusive OR Dn to destination
EORI [4]	BWL	#n,d	-**00	d	-	d	d	d	d	d	d	d	-	-	s	#n XOR d → d	Logical exclusive OR #n to destination
EORI [4]	B	#n,CCR	=====	-	-	-	-	-	-	-	-	-	-	-	s	#n XOR CCR → CCR	Logical exclusive OR #n to CCR
EORI [4]	W	#n,SR	=====	-	-	-	-	-	-	-	-	-	-	-	s	#n XOR SR → SR	Logical exclusive OR #n to SR (Privileged)
EXG	L	Rx,Ry	-----	e	e	-	-	-	-	-	-	-	-	-	-	register ←→ register	Exchange registers (32-bit only)
EXT	WL	Dn	-**00	d	-	-	-	-	-	-	-	-	-	-	-	Dn.B → Dn.W | Dn.W → Dn.L	Sign extend (change .B to .W or .W to .L)
ILLEGAL			-----	-	-	-	-	-	-	-	-	-	-	-	-	PC→-(SSP); SR→-(SSP)	Generate Illegal Instruction exception
JMP		d	-----	-	-	d	-	-	d	d	d	d	d	d	-	↑d → PC	Jump to effective address of destination
JSR		d	-----	-	-	d	-	-	d	d	d	d	d	d	-	PC → -(SP); ↑d → PC	push PC, jump to subroutine at address d
LEA	L	s,An	-----	-	e	s	-	-	s	s	s	s	s	s	-	↑s → An	Load effective address of s to An
LINK		An,#n	-----	-	-	-	-	-	-	-	-	-	-	-	-	An → -(SP); SP → An;	Create local workspace on stack
																SP + #n → SP	(negative n to allocate space)
LSL	BWL	Dx,Dy	***0*	e	-	-	-	-	-	-	-	-	-	-	-		Logical shift Dy, Dx bits left/right
LSR		#n,Dy		d	-	-	-	-	-	-	-	-	-	-	s		Logical shift Dy, #n bits L/R (#n: 1 to 8)
	W	d		-	-	d	d	d	d	d	d	d	-	-	-		Logical shift d 1 bit left/right (.W only)
MOVE [4]	BWL	s,d	-**00	e	s[4]	e	e	e	e	e	e	e	s	s	s[4]	s → d	Move data from source to destination
MOVE	W	s,CCR	=====	s	-	s	s	s	s	s	s	s	s	s	s	s → CCR	Move source to Condition Code Register
MOVE	W	s,SR	=====	s	-	s	s	s	s	s	s	s	s	s	s	s → SR	Move source to Status Register (Privileged)
MOVE	W	SR,d	-----	d	-	d	d	d	d	d	d	d	-	-	-	SR → d	Move Status Register to destination
MOVE	L	USP,An	-----	-	d	-	-	-	-	-	-	-	-	-	-	USP → An	Move User Stack Pointer to An (Privileged)
		An,USP		-	s	-	-	-	-	-	-	-	-	-	-	An → USP	Move An to User Stack Pointer (Privileged)
	BWL	s,d	XNZVC	Dn	An	(An)	(An)+	-(An)	(i,An)	(i,An,Rn)	abs.W	abs.L	(i,PC)	(i,PC,Rn)	#n		

Opcode	Size	Operand	CCR	Effective Address s=source, d=destination, e=either, i=displacement												Operation	Description
	BWL	s,d	XNZVC	Dn	An	(An)	(An)+	-(An)	(i,An)	(i,An,Rn)	abs.W	abs.L	(i,PC)	(i,PC,Rn)	#n		
MOVEA[4]	WL	s,An	-----	s	e	s	s	s	s	s	s	s	s	s	s	s → An	Move source to An (MOVE s,An use MOVEA)
MOVEM[4]	WL	Rn-Rn,d	-----	-	-	d	-	d	d	d	d	d	-	-	-	Registers → d	Move specified registers to/from memory
		s,Rn-Rn		-	-	s	s	-	s	s	s	s	s	s	-	s → Registers	(.W source is sign-extended to .L for Rn)
MOVEP	WL	Dn,(i,An)	-----	s	-	-	-	-	d	-	-	-	-	-	-	Dn → (i,An)...(i+2,An)...(i+4,A.	Move Dn to/from alternate memory bytes
		(i,An),Dn		d	-	-	-	-	s	-	-	-	-	-	-	(i,An) → Dn...(i+2,An)...(i+4,A.	(Access only even or odd addresses)
MOVEQ[4]	L	#n,Dn	-**00	d	-	-	-	-	-	-	-	-	-	-	s	#n → Dn	Move sign extended 8-bit #n to Dn
MULS	W	s,Dn	-**00	e	-	s	s	s	s	s	s	s	s	s	s	±16bit s * ±16bit Dn → ±Dn	Multiply signed 16-bit; result: signed 32-bit
MULU	W	s,Dn	-**00	e	-	s	s	s	s	s	s	s	s	s	s	16bit s * 16bit Dn → Dn	Multiply unsig'd 16-bit; result: unsig'd 32-bit
NBCD	B	d	*U*U*	d	-	d	d	d	d	d	d	d	-	-	-	0 - d_{10} - X → d	Negate BCD with eXtend, BCD result
NEG	BWL	d	*****	d	-	d	d	d	d	d	d	d	-	-	-	0 - d → d	Negate destination (2's complement)
NEGX	BWL	d	*****	d	-	d	d	d	d	d	d	d	-	-	-	0 - d - X → d	Negate destination with eXtend
NOP			-----	-	-	-	-	-	-	-	-	-	-	-	-	None	No operation occurs
NOT	BWL	d	-**00	d	-	d	d	d	d	d	d	d	-	-	-	NOT(d) → d	Logical NOT destination (1's complement)
OR[4]	BWL	s,Dn	-**00	e	-	s	s	s	s	s	s	s	s	s	s[4]	s OR Dn → Dn	Logical OR
		Dn,d		e	-	d	d	d	d	d	d	d	-	-	-	Dn OR d → d	(ORI is used when source is #n)
ORI[4]	BWL	#n,d	-**00	d	-	d	d	d	d	d	d	d	-	-	s	#n OR d → d	Logical OR #n to destination
ORI[4]	B	#n,CCR	=====	-	-	-	-	-	-	-	-	-	-	-	s	#n OR CCR → CCR	Logical OR #n to CCR
ORI[4]	W	#n,SR	=====	-	-	-	-	-	-	-	-	-	-	-	s	#n OR SR → SR	Logical OR #n to SR (Privileged)
PEA	L	s	-----	-	-	s	-	-	s	s	s	s	s	s	-	↑s → -(SP)	Push effective address of s onto stack
RESET			-----	-	-	-	-	-	-	-	-	-	-	-	-	Assert RESET Line	Issue a hardware RESET (Privileged)
ROL	BWL	Dx,Dy	-**0*	e	-	-	-	-	-	-	-	-	-	-	-		Rotate Dy, Dx bits left/right (without X)
ROR		#n,Dy		d	-	-	-	-	-	-	-	-	-	-	s		Rotate Dy, #n bits left/right (#n: 1 to 8)
	W	d		-	-	d	d	d	d	d	d	d	-	-	-		Rotate d 1-bit left/right (.W only)
ROXL	BWL	Dx,Dy	***0*	e	-	-	-	-	-	-	-	-	-	-	-		Rotate Dy, Dx bits L/R, X used then updated
ROXR		#n,Dy		d	-	-	-	-	-	-	-	-	-	-	s		Rotate Dy, #n bits left/right (#n: 1 to 8)
	W	d		-	-	d	d	d	d	d	d	d	-	-	-		Rotate destination 1-bit left/right (.W only)
RTE			=====	-	-	-	-	-	-	-	-	-	-	-	-	(SP)+ → SR; (SP)+ → PC	Return from exception (Privileged)
RTR			=====	-	-	-	-	-	-	-	-	-	-	-	-	(SP)+ → CCR, (SP)+ → PC	Return from subroutine and restore CCR
RTS			-----	-	-	-	-	-	-	-	-	-	-	-	-	(SP)+ → PC	Return from subroutine
SBCD	B	Dy,Dx	*U*U*	e	-	-	-	-	-	-	-	-	-	-	-	Dx_{10} - Dy_{10} - X → Dx_{10}	Subtract BCD source and eXtend bit from
		-(Ay),-(Ax)		-	-	-	-	e	-	-	-	-	-	-	-	-$(Ax)_{10}$ - -$(Ay)_{10}$ - X → -$(Ax)_{10}$	destination, BCD result
Scc	B	d	-----	d	-	d	d	d	d	d	d	d	-	-	-	If cc is true then 1's → d else 0's → d	If cc true then d.B = 11111111 else d.B = 00000000
STOP		#n	=====	-	-	-	-	-	-	-	-	-	-	-	s	#n → SR; STOP	Move #n to SR, stop processor (Privileged)
SUB[4]	BWL	s,Dn	*****	e	s	s	s	s	s	s	s	s	s	s	s[4]	Dn - s → Dn	Subtract binary (SUBI or SUBQ used when
		Dn,d		e	d[4]	d	d	d	d	d	d	d	-	-	-	d - Dn → d	source is #n. Prevent SUBQ with #n.L)
SUBA[4]	WL	s,An	-----	s	e	s	s	s	s	s	s	s	s	s	s	An - s → An	Subtract address (.W sign-extended to .L)
SUBI[4]	BWL	#n,d	*****	d	-	d	d	d	d	d	d	d	-	-	s	d - #n → d	Subtract immediate from destination
SUBQ[4]	BWL	#n,d	*****	d	d	d	d	d	d	d	d	d	-	-	s	d - #n → d	Subtract quick immediate (#n range: 1 to 8)
SUBX	BWL	Dy,Dx	*****	e	-	-	-	-	-	-	-	-	-	-	-	Dx - Dy - X → Dx	Subtract source and eXtend bit from
		-(Ay),-(Ax)		-	-	-	-	e	-	-	-	-	-	-	-	-(Ax) - -(Ay) - X → -(Ax)	destination
SWAP	W	Dn	-**00	d	-	-	-	-	-	-	-	-	-	-	-	bits[31:16]←→bits[15:0]	Exchange the 16-bit halves of Dn
TAS	B	d	-**00	d	-	d	d	d	d	d	d	d	-	-	-	test d→CCR; 1 →bit7 of d	N and Z set to reflect d, bit7 of d set to 1
TRAP		#n	-----	-	-	-	-	-	-	-	-	-	-	-	s	PC→-(SSP);SR→-(SSP); (vector table entry) → PC	Push PC and SR, PC set by vector table #n (#n range: 0 to 15)
TRAPV			-----	-	-	-	-	-	-	-	-	-	-	-	-	If V then TRAP #7	If overflow, execute an Overflow TRAP
TST	BWL	d	-**00	d	-	d	d	d	d	d	d	d	-	-	-	test d → CCR	N and Z set to reflect destination
UNLK		An	-----	-	d	-	-	-	-	-	-	-	-	-	-	An → SP; (SP)+ → An	Remove local workspace from stack
	BWL	s,d	XNZVC	Dn	An	(An)	(An)+	-(An)	(i,An)	(i,An,Rn)	abs.W	abs.L	(i,PC)	(i,PC,Rr)	#n		

Condition Tests (+ OR, ! NOT, ⊕ XOR; * Unsigned, * Alternate cc)					
cc	Condition	Test	cc	Condition	Test
T	true	1	VC	overflow clear	!V
F	false	0	VS	overflow set	V
HI*	higher than	!(C + Z)	PL	plus	!N
LS*	lower or same	C + Z	MI	minus	N
HS*, CC*	higher or same	!C	GE	greater or equal	!(N ⊕ V)
LO*, CS*	lower than	C	LT	less than	(N ⊕ V)
NE	not equal	!Z	GT	greater than	![(N ⊕ V) + Z]
EQ	equal	Z	LE	less or equal	(N ⊕ V) + Z

An Address register (16/32-bit, n=0-7)
Dn Data register (8/16/32-bit, n=0-7)
Rn any data or address register
s Source, **d** Destination
e Either source or destination
#n Immediate data, **i** Displacement
BCD Binary Coded Decimal
↑ Effective address

SSP Supervisor Stack Pointer (32-bit)
USP User Stack Pointer (32-bit)
SP Active Stack Pointer (same as A7)
PC Program Counter (24-bit)
SR Status Register (16-bit)
CCR Condition Code Register (lower 8-bits of SR)
N negative, **Z** zero, **V** overflow, **C** carry, **X** extend
* set according to operation's result, = set directly
- not affected, **0** cleared, **1** set, **U** undefined

1 Long only; all others are byte only
2 Assembler calculates offset
3 Branch sizes: **.B** or **.S** -128 to +127 bytes, **.W** or **.L** -32768 to +32767 bytes
4 Assembler automatically uses A, I, Q or M form if possible. Use #n.L to prevent Quick optimization

Revised by Peter Csaszar, Lawrence Tech University – 2004-2006

NTS-Robotique
QCM

1. Quelle force est utilisée pour faire bouger les nano-robots ?

(a) La force gravitationnelle
(b) La force quantique
(c) La force magnétique

2. Que sont les sites classés Seveso ?

(a) Des sites industriels dangereux
(b) Des lieux de recherche appliquée
(c) Des sites naturels protégés
(d) Des sites militaires protégés

3. Dans quels domaines les robots fixes sont-ils plus efficaces que les humains ?

(a) Le positionnement précis de pièces
(b) La manipulation de pièces fragiles
(c) Les mouvements complexes dans l'espace 3D

4. Quelles sont les domaines de compétences requis pour le traitement des données récupérées par les robots ?

(a) L'intelligence artificielle
(b) Les bases de donnée
(c) La vision par ordinateur
(d) La programmation

5. Quels sont les problèmes rencontrés par les robots lors de l'incident de Fukushima ?

(a) Les batteries des robots avaient une capacité insuffisante
(b) La chaleur des réacteurs a empêché les robots de s'approcher
(c) Les gravats qui ont génés l'accès aux réacteurs
(d) La radioactivité qui réduit la durée de vie des robots

6. Qu'est-ce qui rend difficile la "co-botique" ?

(a) Les émotions humaines
(b) Les codes sociaux
(c) Les risques de blessure
(d) La langue
(e) La proximité avec les humains

7. Les robots tirant leurs aspects et fonctionnements de la nature sont appelés ?

(a) Robots bio-aspirants
(b) Robots bio-inspirés
(c) Robots bio-sensibles
(d) Robots bio-résistants
(e) Robots bio-ressemblants

3 8. **Quelles sont les principales étapes à la maîtrise du travail en équipe des robots ?**

X X (a) Le découpage de tâches complexes en tâches unitaires
X (b) La maîtrise fine des trajectoires
(c) La gestion des recharges différées des robots
X (d) Le positionnement des uns par rapport aux autres
X (e) La synchronisation

3 9. **Pour la géo-localisation, quel problème est posé par le milieu aquatique ?**

(a) Pas d'ondes sonores
(b) Pas d'ondes lumineuses
X X (c) Pas d'ondes radio
(d) Pas d'ondes gamma

40 ~~10~~. **Quel est le nom de l'équipe de Robotique Epita ?**

X X (a) SEAL
(b) TILE
(c) PILE
(d) REAL