ALGO
QCM

1. **Le parcours profondeur d'un graphe est par nature ?**
   (a) Récursif
   (b) Itératif
   (c) Répétitif
   (d) Alternatif

2. **Dans un graphe orienté, s'il existe un circuit $x \rightsquigarrow x$ passant par tous les sommets, le graphe est ?**
   (a) complet
   (b) transitif
   (c) connexe
   (d) fortement connexe

3. **Un graphe orienté de n sommets peut être fortement connexe à partir de ?**
   (a) $n-1$ arcs
   (b) n arcs
   (c) $n+1$ arcs

4. **Deux sommets x et y d'un graphe orienté sont dits adjacents si ?**
   (a) il existe un arc x→y ou un arc y→x
   (b) il existe un arc x→y et un arc y→x
   (c) il existe un chemin x⇝y ou un chemin y⇝x
   (d) il existe un chemin x⇝y et un chemin y⇝x

5. **Le parcours largeur d'un graphe est par nature ?**
   (a) Récursif
   (b) Itératif
   (c) Répétitif
   (d) Alternatif

6. **Dans la forêt couvrante associée au parcours en profondeur d'un graphe orienté G, les arcs x→y tels que x est le père de y sont appelés ?**
   (a) Arcs couvrants
   (b) Arcs en arrière
   (c) Arcs en Avant
   (d) Arcs croisés

7. **Soit un graphe G connexe, sa fermeture transitive est ?**
   (a) Un sous-graphe
   (b) Un graphe partiel
   (c) Un graphe complet

8. **L'algorithme de Warshall est utilisable sur ?**
   - (a) Les graphes orientés statiques
   - (b) Les graphes non orientés statiques
   - (c) Les graphes orientés évolutifs
   - (d) Les graphes non orientés évolutifs

9. **Supposons que *Pref[i]* retourne le Numéro d'ordre préfixe de rencontre d'un sommet i. Lors du parcours en profondeur d'un graphe orienté G, les arcs x→y tels que pref[y] est supérieur à Pref[x] dans la forêt sont appelés ?**
   - (a) Arcs couvrants
   - (b) Arcs en arrière
   - (c) Arcs en Avant
   - (d) Arcs croisés

10. **Calculer la fermeture transitive d'un graphe sert à ?**
   - (a) Déterminer si un graphe est connexe
   - (b) Déterminer les composantes connexes d'un graphe non orienté
   - (c) Déterminer si un graphe est complet

QCM N°5

Lundi 27 novembre 2023

Question 11

Dans $\mathbb{R}^3$, considérons la famille $\mathcal{F} = \big(u_1{=}(1,1,0), u_2{=}(-1,1,1), u_3{=}(-1,1,0)\big)$.

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

b. Cette famille $\mathcal{F}$ est liée

Question 12

Dans $\mathbb{R}^2$, considérons la base canonique $\mathcal{B} = \big(e_1{=}(1,0), e_2{=}(0,1)\big)$, une autre base $\mathcal{B}' = \big(\varepsilon_1{=}(1,-2), \varepsilon_2{=}(2,1)\big)$ et un vecteur $u = (x,y) \in \mathbb{R}^2$.

On note $X = \begin{pmatrix} x \\ y \end{pmatrix}$ et $X' = \begin{pmatrix} x' \\ y' \end{pmatrix}$ les colonnes constituées des coordonnées de u dans les bases $\mathcal{B}$ et $\mathcal{B}'$.

Alors :

a. $\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 1 & -2 \\ 2 & 1 \end{pmatrix} \begin{pmatrix} x' \\ y' \end{pmatrix}$

b. $\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 1 & 2 \\ -2 & 1 \end{pmatrix} \begin{pmatrix} x' \\ y' \end{pmatrix}$

c. $\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 1 & -2 \\ 2 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$

d. $\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 1 & 2 \\ -2 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$

Question 13

Laquelle(Lesquelles) de ces applications est(sont) linéaire(s) ?

a. $f : \left\{ \begin{array}{ccc} \mathbb{R}[X] & \longrightarrow & \mathbb{R}[X] \\ P & \longmapsto & XP^2 \end{array} \right.$

b. $g : \left\{ \begin{array}{ccc} \mathbb{R}[X] & \longrightarrow & \mathbb{R}[X] \\ P & \longmapsto & X^2P \end{array} \right.$

c. $h : \left\{ \begin{array}{ccc} \mathbb{R}[X] & \longrightarrow & \mathbb{R}[X] \\ P & \longmapsto & XP' + P \end{array} \right.$

d. Aucune de ces applications n'est linéaire

Question 14

Soit l'application linéaire $f : \begin{cases} \mathbb{R}[X] & \longrightarrow & \mathbb{R} \\ P & \longmapsto & P(1) \end{cases}$. Alors :

a. $(1, -2, 1) \in \mathrm{Ker}(f)$

b. $(1, -2X, X^2) \in \mathrm{Ker}(f)$

c. $1 - 2X + X^2 \in \mathrm{Ker}(f)$

d. Aucun des autres choix

Question 15

Soit l'application linéaire $f : \begin{cases} \mathbb{R}[X] & \longrightarrow & \mathbb{R} \\ P & \longmapsto & P(1) \end{cases}$. Alors :

a. f est injective, non surjective

b. f est surjective, non injective

c. f n'est ni injective ni surjective

d. f est bijective

Question 16

Soit $A \in \mathcal{M}_3(\mathbb{R})$. On note C_1, C_2 et C_3 ses trois colonnes, L_1, L_2 et L_3 ses trois lignes.

a. On ne change pas $\det(A)$ si on remplace C_2 par $C_1 + C_2 - 2C_3$

b. On ne change pas $\det(A)$ si on remplace C_3 par $C_1 + C_2 - 2C_3$

c. On ne change pas $\det(A)$ si on remplace L_3 par $L_1 - L_2 - L_3$

d. On ne change pas $\det(A)$ si on remplace L_1 par $L_1 - L_2 - L_3$

e. Aucun des autres choix

Question 17

Considérons deux matrices A et B dans $\mathcal{M}_3(\mathbb{R})$ et $\lambda \in \mathbb{R}$. Alors :

a. $\det(A + B) = \det(A) + \det(B)$

b. $\det(\lambda A) = \lambda \det(A)$

c. $\det(\lambda A) = \lambda^2 \det(A)$

d. $\det(A \times B) = \det(A) \times \det(B)$

e. Aucun des autres choix

Question 18

Soient E un $\mathbb{R}$-ev, $f \in \mathcal{L}(E)$ et $\lambda \in \mathbb{R}$. Le réel λ est valeur propre de f si et seulement si :

a. $\exists u \in E, f(u) = \lambda u$

b. $\exists u \in E, f(\lambda u) = u$

c. $\exists u \in E, f(u) = \lambda u$ et $u \neq 0_E$

d. $\exists u \in E, f(\lambda u) = u$ et $u \neq 0_E$

e. Aucun des autres choix

Question 19

Soient E un $\mathbb{R}$-ev, $f \in \mathcal{L}(E)$ et $\lambda \in \mathbb{R}$. On note id l'application identité de E. Le réel λ est valeur propre de f si et seulement si :

a. $\text{Ker}(f - \lambda id) = \{0_E\}$

b $\text{Ker}(f - \lambda id) \neq \{0_E\}$

c. $\text{Im}(f - \lambda id) = \{0_E\}$

d. $\text{Im}(f - \lambda id) \neq \{0_E\}$

e. Aucun des autres choix

Question 20

Soit $A \in \mathcal{M}_3(\mathbb{R})$. On note I la matrice identité de $\mathcal{M}_3(\mathbb{R})$. Le polynôme caractéristique de A est :

a. $P_A(X) = \det(XA + I)$

b. $P_A(X) = \det(XA - I)$

c. $P_A(X) = \det(A + XI)$

d. $P_A(X) = \det(XA)$

e. Aucun des autres choix

QCM 7 Azar Chap20 S3 (pg 446 ex44) fall 23

Choose the **one** correct answer for each question.

21. Epita does not have air conditioning, but I wish it ____ some good fans, at least.
a. will have
b. would have
c. had
d. has

22. Afshari didn't come to the party on Saturday. Disappointed, you say: I wish ____
a. she had come.
b. she came.
c. she would have come.
d. she will come.

23. It's raining! I wish the sun ____ right now.
a. shined
b. were shining
c. shines
d. would be shining

24. I don't know how to play the guitar. I wish I ____ how to play it!
a. knew
b. will know
c. had known
d. know

25. Pedro forgot to write down his password. He wishes he _____ to write down his password.
a. didn't forget
b. had forgotten
c. had not forgotten
d. would not forget

26. Susan didn't eat dinner before she went to bed. She wasn't hungry then, but she was at 2 in the morning. She wishes she ___ dinner.
a. had eaten
b. had as usual
c. would have ate
d. ate

27. Declan did not come to Prologin. I wish he _____ to Prologin.
a. had come
b. would come
c. come
d. came

28. Michael does not like being a truck driver. He wishes he ___ a taxicab driver instead.
a. became
b. had become
c. has been
d. would be

29. Michael (the truck driver) wishes he ____ people around in his car instead of furniture.
a. had driven
b. drove
c. has been driving
d. drives

30. Maryam can't afford to come on holiday with us but I wish she _____ , because I think we would have a great time together.
a. will come
b. would come
c. can come
d. None of the above.

QCM 7 – OC S3 2023/24 (Week 27 November)

31. Which of the following is **NOT** a stage of Oberg's Culture Shock theory?

a) Reverse Shock
b) Honeymoon
c) Adaptation
d) Adjustment

32. Which of the following according to Oberg are the negative aspects of culture shock? *Choose all that apply*

a) Stress
b) Anxiety
c) Hair loss
d) Feeling of helplessness

33. Which is the 3rd stage of Oberg's Culture Shock theory?

a) Reserve Shock
b) Honeymoon
c) Adaptation
d) Adjustment

34. The repeated stages of culture shock that occurs on returning home is illustrated by which theory?

a) U Curve
b) W Curve
c) Adjustment
d) Adaptation

35. We are more likely to experience difficulties that are manageable during the ***Adjustment*** phase of Oberg's stages of culture shock. True or False?

a) True
b) False

36. What is the ***cultural iceberg*** discussed in the 'Why you'll hate living in Japan' video composed of? *Choose all that apply.*

a) Material Aspects
b) Culture Shock
c) Non-Material Aspects
d) Employment options

37. **[Video]** Mr Eats states the length of time spent in The Honeymoon Phase is the only phase that is similar for everybody. True or False?

a) True
b) False

38. **[Video]** According to Mr Eats when you become more comfortable getting help from others and improving your language skills, you are at the _____ phase of culture shock.

a) Adaptation
b) Delicate
c) Adjustment
d) Honeymoon

39. **[Video]** What are some of the problems Mr Eats states that people can encounter in Japan? *Choose all that apply*

a) Impolite locals
b) Difficult work culture
c) Limited salary options
d) Loneliness

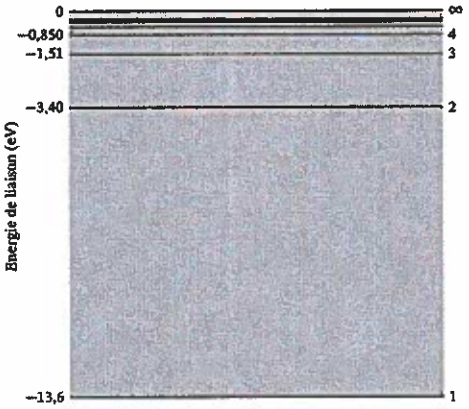
40. **[Video]** What advice is given to help overcome Culture Shock? *Choose all that apply*

a) Accept your culture shock
b) Get more comfortable with the language
c) Do an activity that makes you happy
d) Don't expect too much from the country

QCM Physique – InfoS3 – 27.11

Pensez à bien lire les questions ET les réponses proposées (attention à la numérotation des réponses)

Les Q41 à Q43. s'appuient sur le diagramme d'énergie ci-dessous de l'atome d'hydrogène de Bohr.



n=1 ; $E_1 = -13{,}6\ eV$

n=2 ; $E_2 = -3{,}4\ eV$

n=3 ; $E_3 = -1{,}51\ eV$

n=4 ; $E_4 = -0{,}850\ eV$

n=5 ; $E_5 = -0{,}54\ eV$

n=6 ; $E_6 = -0{,}37\ eV$

n->+∞ ; $E_\infty = 0$

1 ev = $1{,}6.10^{-19}$ J

Q41. La variation d'énergie en jeu pour passer de l'orbite n = 4 à l'orbite n' = 2 est égale à :

a. -3,4 ev
b. -0,850 ev
c. -2,55 ev
d. 2,55 ev

Q42. Pour arracher un électron à l'atome (ionisation) depuis l'état fondamental vers l'état $n \to +\infty$, la variation d'énergie mise en jeu est :

a. 13,6 ev
b. -13,6 ev
c. 0 ev
d. 1 ev

Q43. L'on apporte à l'électron dans l'état n = 2 un quanta d'énergie 1 ev. Que se passe-t-il ?

a. L'électron passe à l'état d'énergie supérieure n = 3.
b. L'électron passe à l'état d'énergie inférieure n = 1.
c. Il ne se passe rien.
d. L'électron est arraché à l'atome.

Q44. Le principe d'incertitude d'Heisenberg pour une particule de masse constante m, de vitesse v (donc de quantité de mouvement p = mv), repérée par sa position x, a pour expression ($\hbar$ désigne la constante de Planck réduite, et pour une grandeur a, Δa désigne son incertitude) :

a. $\Delta x \Delta p \geq \frac{\hbar}{2}$
b. $\Delta x \Delta v \geq \frac{\hbar}{2}$
c. $m.\Delta x \Delta v \geq \frac{\hbar}{2}$
d. $m.\Delta x \Delta v = \frac{\hbar}{2}$

Q45. Le principe d'incertitude d'Heisenberg signifie que, pour une particule de masse constante m :

a. Si l'on connaît sa position avec une grande précision, alors on ne peut pas connaître précisément sa vitesse.
b. Si l'on connaît sa vitesse avec une grande précision, alors on ne peut pas connaître précisément sa position.
c. Si l'on connaît sa position avec une grande précision, alors on connaît également sa vitesse avec une grande précision.
d. Aucune de ces réponses.

Q46. Le principe d'incertitude d'Heisenberg a des effets négligeables sur :

a. Les objets ayant une faible vitesse
b. Les objets ayant une grande vitesse
c. Les objets macroscopiques
d. Les objets microscopiques

Q47. $\mathcal{H}$ désigne l'opérateur hamiltonien, et E l'énergie d'une particule. L'équation de Schrödinger indépendante du temps, appliquée à la fonction d'onde ψ de cette particule a pour expression :

a. $\mathcal{H}\psi = \psi$
b. $\mathcal{H}\psi = E^2\psi$
c. $\mathcal{H}\psi = E\psi^2$
d. $\mathcal{H}\psi = E\psi$

Q48. ψ désigne la fonction d'onde d'une particule, définie pour $x \in \mathbb{R}$ (cas unidimensionnel). La densité de probabilité de présence de la particule s'obtient en calculant :

a. $|\psi(x)|$
b. $|\psi(x)|^2$
c. $\left|\frac{\partial^2\psi(x)}{\partial x^2}\right|$
d. $\left|\frac{\partial^2\psi(x)}{\partial x^2}\right|^2$

Q49. L'intégrale de la densité de probabilité de présence sur l'intervalle $]-\infty ; +\infty[$ est :

a. Egale à 0
b. Egale à 1
c. Strictement inférieure à 1
d. Aucune de ces réponses

Q50. On donne la fonction d'onde ψ associée à une particule : $\psi(x) = \psi_0 e^{-\frac{x}{x_0}}$; où x_0 et ψ_0 sont des constantes. On a :

a. $\frac{d^2\psi(x)}{dx^2} = \psi_0 x_0^2 e^{-\frac{x}{x_0}}$
b. $\frac{d^2\psi(x)}{dx^2} = \psi_0 x_0 e^{-\frac{x}{x_0}}$
c. $\frac{d^2\psi(x)}{dx^2} = \frac{\psi_0}{x_0^2} e^{-\frac{x}{x_0}}$
d. $\frac{d^2\psi(x)}{dx^2} = -\frac{\psi_0}{x_0^2} e^{-\frac{x}{x_0}}$

QCM 7
Architecture des ordinateurs

Lundi 27 novembre 2023

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

51. 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)`
    C. Aucune de ces réponses.
    D. `MOVEM.L A4/A5/D1/D2/D3,-(A7)`

52. Soient les deux instructions suivantes :
```
CMP.W D1,D2
BLE   NEXT
```
    Branchement à `NEXT` si :
    A. D1 = $92181892 et D2 = $18929218
    B. D1 = $18929218 et D2 = $18929218
    C. D1 = $92181892 et D2 = $92181892
    D. D1 = $18929218 et D2 = $92181892

53. Quelle(s) instruction(s) n'est (ne sont) pas possible(s) ?
    A. SUBI.B #2,(A2)
    B. SUBI.L #42,D0
    C. SUBI.L D2,D3
    D. SUBI.L #8,A2

54. Quelle(s) instruction(s) n'est (ne sont) pas possible(s) ?
    A. MOVEQ.W #42,D0
    B. MOVEQ.L D1,D0
    C. MOVEQ.L #42,D0
    D. MOVEQ.B #42,D0

55. Quelle(s) instruction(s) n'est (ne sont) pas possible(s) ?
    A. MOVEA.B #50,D0
    B. MOVEA.B #50,A0
    C. MOVEA.L #50,D0
    D. MOVEA.L #50,A0

56. Trouvez le ou les nombres manquants pour l'addition sur 16 bits suivante afin d'obtenir la bonne combinaison de *flags* : `$609A + $?` avec **N** = 1, **Z** = 0, **V** = 1, **C** = 0
    A. $1F66
    B. $7000
    C. $A000
    D. Aucune de ces réponses

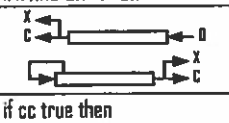
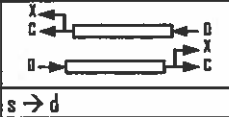
57. Trouvez le ou les nombres manquants pour l'addition sur 8 bits suivante afin d'obtenir la bonne combinaison de *flags* : `$71 + $?` avec **N** = 0, **Z** = 1, **V** = 0, **C** = 0
    A. $00
    B. $0F
    C. $8F
    D. Aucune de ces réponses

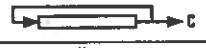
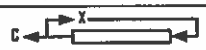
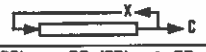
58. Quelle est la valeur de D1.L après l'exécution de l'instruction suivante ? `SUB.B D0,D1`
    Valeurs initiales : D0.L = $00000007, D1.L = $00000002
    A. $00000005
    B. $000000FB
    C. $FFFFFFFB
    D. Aucune de ces réponses

59. Quelle est la valeur de D1.L après l'exécution de l'instruction suivante ? `ADD.B D0,D1`
    Valeurs initiales : D0.L = $000001F0, D1.L = $00000111
    A. $00000301
    B. $00000201
    C. $00000101
    D. Aucune de ces réponses

60. Quelle est la valeur de D1.L après l'exécution de l'instruction suivante ? `SUB.W D0,D1`
    Valeurs initiales : D0.L = $00000007, D1.L = $00000002
    A. $0000FFFB
    B. $0000FFF5
    C. $000000FB
    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	-	-	-	-	-	-	-	-	-	-	-	$Dy_{10} + Dx_{10} + X \rightarrow Dx_{10}$	Add BCD source and eXtend bit to
		-(Ay),-(Ax)		-	-	-	-	e	-	-	-	-	-	-	-	$-(Ay)_{10} + -(Ax)_{10} + X \rightarrow -(Ax)_{10}$	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	If cc true then d.B = 11111111
																else 0's → d	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);	Push PC and SR, PC set by vector table #n
																(vector table entry) → PC	(#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,Rn)	#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

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

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

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