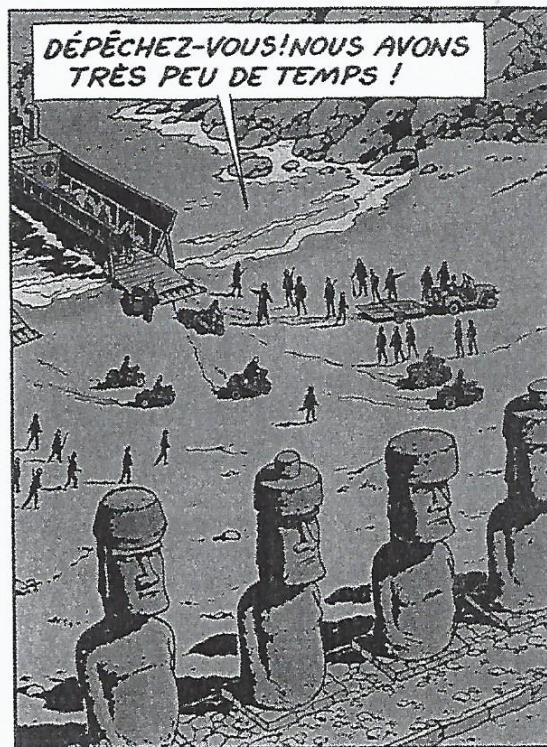


ALGO  
QCM

1. Deux sommets d'un graphe non orienté sont dits adjacents si ?
  - (a) il existe deux arcs les joignant
  - (b) le graphe est complet
  - (c) ils ont au moins une extrémité commune
  - (d) s'il existe une arête les joignant
2. Dans un graphe non orienté, une chaîne dont toutes les arêtes sont distinctes deux à deux et telle que les deux extrémités coïncident est ?
  - (a) un circuit
  - (b) un cycle
  - (c) connexe
  - (d) fortement connexe
  - (e) un chemin
3. Un graphe partiel  $G'$  de  $G = \langle S, A \rangle$  est défini par ?
  - (a)  $\langle S, A' \rangle$  avec  $A' \subseteq A$
  - (b)  $\langle S', A \rangle$  avec  $S' \subseteq S$
  - (c)  $\langle A, S \rangle$
4. Dans un graphe orienté, on dit que l'arc  $U = y \rightarrow x$  est ?
  - (a) incident à  $x$  vers l'extérieur
  - (b) accident à  $x$  vers l'extérieur
  - (c) incident à  $x$  vers l'intérieur
  - (d) accident à  $x$  vers l'intérieur
5. Dans un graphe non orienté, s'il existe une arête  $x - y$  pour tout couple de sommet  $\{x, y\}$  le graphe est ?
  - (a) complet
  - (b) partiel
  - (c) parfait
  - (d) connexe
6. Deux arcs d'un graphe orienté sont dits adjacents si ?
  - (a) il existe deux arcs les joignant
  - (b) le graphe est complet
  - (c) ils ont au moins une extrémité commune

7. Dans un graphe non orienté  $G = \langle S, A \rangle$ , Le sous-graphe connexe maximal  $G' = \langle S', A' \rangle$  est une composante connexe du graphe  $G$  ?  
 (a) vrai  
 (b) faux
8. Dans un graphe valué  $G = \langle S, A, C \rangle$ , les coûts sont portés par ?  
 (a) les relations  
 (b) les sommets
9. Un chemin qui ne contient pas plusieurs fois un même sommet est ?  
 (a) élémentaire  
 (b) optimal  
 (c) plus court  
 (d) une chaîne
10. Dans un graphe orienté, s'il existe un chemin  $x \rightsquigarrow x$  passant par tous les sommets du graphe le graphe est ?  
 (a) complet  
 (b) partiel  
 (c) parfait  
 (d) fortement connexe



## QCM N°7

lundi 28 novembre 2016

### Question 11

Soient  $(A, B) \in \mathcal{M}_n^2(\mathbb{R})$  (où  $n \geq 2$ ) et  $\lambda \in \mathbb{R}$ . Alors

- a.  $\det(A + B) = \det(A) + \det(B)$
- b.  $\det(AB) = \det(A)\det(B)$
- c.  $\det(\lambda A) = \lambda \det(A)$
- d. si  $A$  est diagonale, alors  $\det(A) = \text{tr}(A)$
- e. rien de ce qui précède

### Question 12

Soit  $A \in \mathcal{M}_n(\mathbb{R})$  (où  $n \geq 2$ ) telle que  $A^2 = A - I_n$  où  $I_n$  est la matrice identité d'ordre  $n$ . Alors

- a.  $X^2 - 1$  est un polynôme annulateur de  $A$
- b.  $X^2 - X + 1$  est un polynôme annulateur de  $A$
- c.  $X^3 - X^2 + X$  est un polynôme annulateur de  $A$
- d.  $X^2 - X$  est un polynôme annulateur de  $A$ .
- e. rien de ce qui précède

### Question 13

Soient  $A \in \mathcal{M}_n(\mathbb{R})$  (où  $n \geq 2$ ) et  $\lambda$  une valeur propre de  $A$ . Alors en notant  $I_n$  la matrice identité d'ordre  $n$

- a.  $\text{Ker}(A - \lambda I_n) \neq \{0\}$
- b.  $A - \lambda I_n$  est inversible
- c.  $\exists X \in \mathcal{M}_{n,1}(\mathbb{R}) \setminus \{0\}, \quad AX = \lambda X$
- d. rien de ce qui précède

### Question 14

Soient  $A \in \mathcal{M}_2(\mathbb{R})$ ,  $\lambda$  et  $\mu$  deux valeurs propres réelles distinctes de  $A$ . Alors

- a.  $\mathbb{R}^2 = E_\lambda \oplus E_\mu$
- b.  $E_\lambda$  et  $E_\mu$  sont en somme directe
- c.  $A$  est diagonalisable
- d. rien ce de qui précède

### Question 15

Soient  $E$  un  $\mathbb{R}$ -ev,  $u \in \mathcal{L}(E)$ ,  $\lambda$  une valeur propre de  $u$ . Alors  $x \in E_\lambda$  signifie

- a.  $u(\lambda x) = \lambda u(x)$
- b.  $u(x) = \lambda x$
- c.  $u(x) - \lambda x \neq 0$
- d.  $x \in \text{Im}(u - \lambda id)$
- e. rien de ce qui précède

### Question 16

Soit  $A = \begin{pmatrix} 1 & 0 & 2 \\ 0 & 2 & 0 \\ 2 & 0 & 1 \end{pmatrix}$ . Alors le polynôme caractéristique de  $A$  est

- a.  $(1 - X)(2 - X)^2$
- b.  $(1 - X)^2(2 - X)$
- c.  $(2 - X)(X - 1)(X - 3)$
- d.  $(2 + X)(X + 1)(X - 3)$
- e. rien de ce qui précède

### Question 17

Soit  $A = \begin{pmatrix} 0 & 1 & 0 \\ -1 & 2 & 0 \\ 3 & 2 & 2 \end{pmatrix}$ . Alors

- a.  $-2$  est une valeur propre de  $A$
- b.  $1$  est une valeur propre de  $A$
- c.  $2$  est une valeur propre de  $A$
- d.  $-1$  est une valeur propre de  $A$
- e. rien de ce qui précède

### Question 18

Soit  $A = \begin{pmatrix} 1 & 2 \\ -1 & 4 \end{pmatrix}$ . Alors

- a.  $\begin{pmatrix} 2 \\ 1 \end{pmatrix}$  est un vecteur propre associé à la valeur propre  $2$
- b.  $\begin{pmatrix} 2 \\ -1 \end{pmatrix}$  est un vecteur propre associé à la valeur propre  $3$
- c.  $\begin{pmatrix} 1 \\ -1 \end{pmatrix}$  est un vecteur propre associé à la valeur propre  $3$
- d.  $\begin{pmatrix} 1 \\ 1 \end{pmatrix}$  est un vecteur propre associé à la valeur propre  $3$
- e. rien de ce qui précède

### Question 19

Soient  $E = \mathbb{R}_3[X]$  et  $F = \text{Vect}(\{1 - X, 1 + X, 1 - X^2, 1 - X^3\})$ . Alors

- a.  $F$  est un sev de  $E$
- b. La famille  $(1 - X, 1 + X, 1 - X^2, 1 - X^3)$  est libre
- c.  $\dim(F) = \dim(E)$
- d.  $F = E$
- e. rien de ce qui précède

### Question 20

Soient  $E$  un  $\mathbb{R}$ -ev et  $(f, g) \in (\mathcal{L}(E))^2$  quelconque. Alors

- a.  $\text{Ker}(g) \subset \text{Ker}(g \circ f)$
- b.  $\text{Ker}(g \circ f) \subset \text{Ker}(f)$
- c.  $\text{Im}(f) \subset \text{Im}(g \circ f)$
- d.  $\text{Im}(g \circ f) \subset \text{Im}(f)$
- e. rien de ce qui précède

Choose the correct answer(s).

21. My mom knows a singer \_\_\_\_ last name is Singer.

- a. which
- b. of who
- c. whose
- d. whom

22. The man \_\_\_\_ the Americans elected for president has never held public office.

- a. whom
- b. whose ↗
- c. which ↗
- d. leave it blank
- e. A and D

23. The professor....

- a. who's the course I am taking is excellent.
- b. whose the course I am taking is excellent.
- c. that the course I am taking is excellent.
- d. whose course I am taking is excellent.

In 24-30, the two sentences have been combined for you. Which is the correct logical combination?  
(Punctuation is taken into account.)

24. The people were very nice. We visited their house yesterday.

- a. The people whom house we visited them yesterday, were very nice. ↗
- b. The people whose house we visited yesterday were very nice.
- c. The people whose the house we visited yesterday were very nice. ↗
- d. The people whose their house we visited was very nice. ↗

25. The city was beautiful. We spent our vacation there.

- a. The city in where we spent our vacation was beautiful. ↗
- b. The city which we spent our vacation was beautiful. ↗
- c. The city where we spent our vacation was beautiful.
- d. None of the above.

26. August is the month. The weather is usually the hottest then (in that month).

- a. August is the month where it is usually the hottest. ↗
- b. August is the month when the weather is usually the hottest.
- c. August is the month when is usually the hottest. ↗
- d. August is the month on which the weather is usually the hottest.

27. The school was destroyed in a fire ten years ago. I went to school there.

- a. The school I went was destroyed in a fire ten years ago.
- b. The school when I went to was destroyed in a fire ten years ago.
- c. The school I went to was destroyed in a fire ten years ago.
- d. The school to where I went to was destroyed in a fire ten years ago.

28. Suzanne Vega teaches singing to a class of students. Their native language is not English.

- a. Suzanne Vega teaches singing to a class of students that their native language is not English. ↗

- b. Suzanne Vega teaches singing to a class of students whom their native language is not English.
- c. Suzanne Vega teaches singing to a class of students whose native language is not English.
- d. Suzanne Vega teaches singing to a class of students their native language is not English.

29. The man is standing over there. Anne brought him to the party.

- a. The man standing over there is whom Anne brought to the party.
- b. That is the man whom Anne brought to the party is standing over there.
- c. That is the man whose Anne brought to the party, standing over there.
- d. None of the above.

30. Did you read about the candidate? He is accused of tax evasion.

- a. Did you read about the candidate whom is accused of tax evasion?
- b. Did you read about the candidate that is accused of tax evasion?
- c. Did you read about the candidate whose accused of tax evasion?
- d. All of the above.

31. The 'father' of French anthropology is considered to be:  
a. Michel Foucault  
b. Bruno Latour  
c. Claude Lévi-Strauss  
d. Pierre Bourdieu
32. The oral presentations provide you with what kind of anthropological information?  
a. social and cultural characteristics of a country  
b. comparisons between cultural characteristics of different countries  
c. different worldviews  
d. all of the above
33. The model of culture associated with Lévi-Strauss is called:  
a. structuralism  
b. cultural evolution  
c. cultural relativism  
d. fundamentalism
34. Learning another person's worldview asks you to become:  
a. less ethnocentric  
b. less culturally relativist  
c. more ethnocentric  
d. less reflexive
35. Michel Foucault argued that we can understand structures of power in society through analyzing different:  
a. worldviews  
b. discourses  
c. theories of culture  
d. habitus
36. "Cultural Relativism" refers to the principle that:  
a. how a person acts or believes can be understood in the framework of his/her own culture  
b. all cultures are related  
c. some cultures are primitive relative to others  
d. cultures are difficult to study
37. The 'science' in *science humaine/social science* refers to:  
a. using a rigorous methodology to investigate theories about culture and society  
b. using popular understandings about culture to support scientific theories  
c. studying human beings using the methods to study bacteria in a laboratory  
d. 'science' in this phrase is just a joke
38. The concept of Weltanschauung sees culture and \_\_\_\_\_ as closely interacting in how a person views the world.  
a. economy  
b. language  
c. music  
d. technology
39. Structuralism was influenced by theories about human \_\_\_\_\_.  
a. behavior  
b. psychology  
c. genetics  
d. language

40. Modern theories about culture see culture as primarily an aspect of human \_\_\_\_\_.  
a. artistic spirit  
b. beauty  
c. cognition (thinking)  
d. unconscious desires

Q.C.M n°7 de Physique

41- On considère un plan (xOy) infini chargé uniformément. Le champ électrique sera :

- a)  $E(x)$       b)  $E(y)$       c)  $E(x, y)$       d)  $E(z)$

42- Un élément de volume  $dV$  situé en un point P avec une distribution de charges  $\rho(P)$  crée un champ électrique en un point M avec  $\vec{u}$  unitaire :

a)  $d\vec{E}(M) = \frac{k \cdot \rho(P) \cdot dV}{PM^2} \vec{u}$       b)  $d\vec{E}(M) = \frac{k \cdot \rho(P) \cdot dV}{PM} \vec{u}$       c)  $d\vec{E}(M) = \frac{k \cdot \rho(P) \cdot dV}{PM^2}$

43- Un disque de rayon  $R$  d'axe (Oz) chargé uniformément de densité  $\sigma$  crée en un point M ( $z > 0$ ) un champ électrique  $E(M) = \frac{\sigma}{2\epsilon_0} \left( 1 - \frac{z}{(R^2+z^2)^{\frac{1}{2}}} \right)$ . À partir de cette expression on retrouve le champ électrique créé par le plan (xOy) infini chargé de la question 41 :

a)  $\vec{E}(M) = \frac{\sigma}{2\epsilon_0} \left( 1 - \frac{z}{(R^2+z^2)^{\frac{1}{2}}} \right) \vec{u}_z$       b)  $\vec{E}(M) = \frac{\sigma}{2\epsilon_0} \vec{u}_z$       c)  $\vec{E}(M) = \frac{\sigma}{2\epsilon_0} \vec{u}_r$

44- Le champ créé par un anneau, chargé uniformément et d'axe (Oz), en un point M situé sur cet axe est :

- a) nul      b) radial      c) suivant (Oz)

45- Le champ créé par une sphère chargée avec une densité surfacique constante est :

- a) quelconque      b) radial      c) suivant  $\vec{u}_\varphi$

46- Un fil infini et uniformément chargé crée un vecteur champ électrique en un point M extérieur au fil de direction radiale. En utilisant le théorème de Gauss, quelle surface choisir ?

- a) un cylindre      b) une sphère      c) un cône

47- Dans le théorème de Gauss apparaît la charge  $Q_{int}$ . Où se situe cette charge ?

- a) dans n'importe quel volume  
b) sur la surface de Gauss  
c) dans l'espace intérieur délimité par la surface de Gauss

48- Dans le théorème de Gauss, le vecteur élément de surface  $\vec{dS}$  doit être :

- a) perpendiculaire à la surface de Gauss et orienté vers l'intérieur de cette surface
- b) incliné par rapport à la normale de la surface de Gauss
- c) tangent à la surface de Gauss
- d) perpendiculaire à la surface de Gauss et orienté vers l'extérieur

49- En considérant la distribution sphérique de charge suivante  $\rho(r) = \rho_0 \left(1 - a \frac{r^2}{R^2}\right)$  où  $a$  et  $\rho_0$  sont des constantes, quel est le champ électrique créé en un point M extérieur à la boule de rayon  $R$  :

a)  $\vec{E}(M) = \vec{0}$        b)  $\vec{E}(M) = \frac{kQ_{int}}{r^2} \vec{u}_r$       c)  $\vec{E}(M) = \rho_0 \left(1 - a \frac{r^2}{R^2}\right) \vec{u}_r$

50- Le champ  $\vec{E}(M)$  créé par un cylindre creux infini et de rayon  $a$  chargé uniformément en surface en un point M situé à l'intérieur de celui-là est

a)  $\vec{E}(M) = \vec{0}$       b) non nul mais constant      c)  $\vec{E}(M) = k\sigma \frac{a}{r} \vec{u}_r$

# QCM Electronique – InfoS3

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

## Diode Zéner

**Q1.** En polarisation directe, on peut représenter la diode Zéner à l'aide de l'un des 2 modèles : à seuil ou linéaire – le modèle idéal n'existant pas pour cette diode.

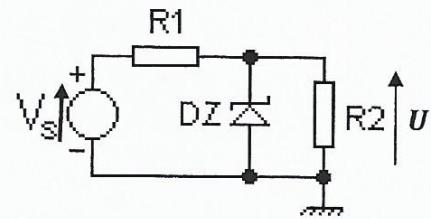
✓ a- VRAI

✗ b- FAUX

Soit le montage ci-contre pour les questions de 2 à 5 :

**Q2.** Choisir l'affirmation correcte :

- a- La diode est polarisée en direct.
- b- La diode est bloquée quelque soit la valeur de la tension  $V_S$ .
- c- Lorsque la diode est bloquée, la résistance  $R_2$  est court-circuitée.
- ✗ d- Lorsque la diode Zéner est passante (en inverse), la tension à ses bornes est quasiment constante tant que le courant qui la traverse reste, en valeur absolue, inférieur à une valeur limite spécifiée par le composant.



**Q3.** La diode DZ est passante en inverse si : (choisir l'affirmation correcte)

- a-  $-|V_Z| < U < V_0$
- b-  $-V_0 < U < |V_Z|$
- ✗ c-  $U \leq -|V_Z|$
- ✗ d-  $U \geq |V_Z|$

**Q4.** La diode DZ est bloquée si : (choisir l'affirmation correcte)

- a-  $-|V_Z| < U < V_0$
- ✗ b-  $-V_0 < U < |V_Z|$
- c-  $U \leq -V_0$
- d-  $U \leq -|V_Z|$

**Q5.** Lorsque la diode DZ est passante en inverse, que vaut la tension U ? (en utilisant le modèle à seuil)

- a-  $U = -|V_Z|$
- ✓ b-  $U = -V_0$
- ✗ c-  $U = |V_Z|$
- d-  $U = \frac{R_2}{R_2+R_1} V_S$

## Transistor bipolaire

**Q6.** Le transistor bipolaire

- a- Peut se modéliser à l'aide de 3 diodes
- b- Est un composant à 3 électrodes comportant 2 jonctions PN
- c- Comporte trois segments dopés de manière identique
- d- Est un composant à 2 électrodes comportant 3 jonctions PN

**Q7.** L'effet transistor :

- a- Permet de faire passer un grand courant entre l'émetteur et le collecteur.
- b- Permet de faire passer un grand courant entre la base et le collecteur.
- c- Permet de faire passer un grand courant entre l'émetteur et la base.

**Q8.** Lorsque l'on fait fonctionner le transistor comme un interrupteur (2 réponses):

- a- Le transistor est équivalent à un interrupteur fermé lorsqu'un courant passe dans la base.
- b- Le transistor est équivalent à un interrupteur fermé lorsqu'aucun courant ne passe dans la base.
- c- Le transistor est équivalent à un interrupteur ouvert lorsqu'aucun courant ne passe dans la base.
- d- Le transistor est équivalent à un interrupteur ouvert lorsqu'un courant passe dans la base.

**Q9.** La caractéristique  $I_B = f(V_{BE})$  est identique à celle :

- a- D'un générateur de tension
- b- D'un générateur de courant
- c- D'une diode
- d- De  $I_C = f(V_{CE})$

**Q10.** La caractéristique  $I_C = f(V_{CE})$  varie en fonction de  $I_B$  :

- a- Vrai
- b- Faux

# QCM 7

## Architecture des ordinateurs

Lundi 28 novembre 2016

11. Soit l'instruction suivante : MOVE.L #\$5C48,D0. Que représente la valeur \$5C48 ?
  - A. Une adresse sur 16 bits.
  - B. Une donnée immédiate sur 16 bits.
  - C. Une adresse sur 32 bits.
  - D. Une donnée immédiate sur 32 bits.
  
12. Soit l'instruction suivante : MOVE.W \$5C48,D0. Que représente la valeur \$5C48 ?
  - A. Une adresse sur 16 bits.
  - B. Une donnée immédiate sur 16 bits.
  - C. Une adresse sur 32 bits.
  - D. Une donnée immédiate sur 32 bits.
  
13. Le bus d'adresse du 68000 est de :
  - A. 24 bits
  - B. 32 bits
  - C. 64 bits
  - D. 16 bits
  
14. Soit l'instruction suivante : MOVE.W (A0)+,D0
  - A. A0 ne change pas.
  - B. A0 est incrémenté de 1.
  - C. A0 est incrémenté de 2.
  - D. A0 est incrémenté de 4.
  
15. Soit l'instruction suivante : MOVE.W 2(A0),D0
  - A. A0 ne change pas.
  - B. A0 est incrémenté de 1.
  - C. A0 est incrémenté de 2.
  - D. A0 est incrémenté de 4.

16. Quels modes d'adressage ne spécifient pas d'emplacement mémoire ? (deux réponses)

- A. Mode d'adressage direct.
- B. Mode d'adressage indirect.
- C. Mode d'adressage absolu.
- D. Mode d'adressage immédiat.

17. Soient les deux instructions suivantes :

TST.B D0  
BMI NEXT

L'instruction BMI effectue le branchement si :

- A. D0 = \$FF
- B. D0 = \$00
- C. D0 = \$50
- D. D0 = \$7F

18. Soient les deux instructions suivantes :

CMP.L D1,D2  
BGT NEXT

L'instruction BGT effectue le branchement si :

- A. D1 > D2 (comparaison signée)
- B. D2 > D1 (comparaison non signée)
- C. D2 > D1 (comparaison signée)
- D. D1 > D2 (comparaison non signée)

19. Si **D0** = \$FFFF45BC et **D1**=\$FFFF7B44, quelles sont les valeurs des *flags* après l'instruction suivante ? ADD.W D0,D1

- A. N = 1, Z = 0, V = 0, C = 1
- B. N = 1, Z = 0, V = 1, C = 1
- C. N = 1, Z = 0, V = 1, C = 0
- D. N = 0, Z = 0, V = 1, C = 0

20. Les étapes pour empiler une donnée sont :

- A. Décrémenter A7 puis écrire la donnée dans (A7).
- B. Écrire la donnée dans (A7) puis décrémenter A7.
- C. Lire la donnée dans (A7) puis incrémenter A7.
- D. Incrémenter A7 puis lire la donnée dans (A7).

| EASy68K Quick Reference v1.8 |                 |                         |   |   |                                  |                   |                   |                 |  | <a href="http://www.wowgwep.com/EASy68K.htm">http://www.wowgwep.com/EASy68K.htm</a> |  |  |             | Copyright © 2004-2007 By: Chuck Kelly |  |
|------------------------------|-----------------|-------------------------|---|---|----------------------------------|-------------------|-------------------|-----------------|--|---|--|--|-------------|---------------------------------------|--|
| 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 -(Ay),-(Ax)       | *U**U*                                  | e - - - - - - - -   | e - - - - - - - -                | - - - - - - - -   | - - - - - - - -   | - - - - - - - - | Dy + Dx + X → Dx <sub>0</sub> -(Ay) <sub>0</sub> + -(Ax) <sub>0</sub> + X → -(Ax) <sub>0</sub> |   | Add BCD source and eXtend bit to destination, BCD result |  |             |                                       |  |
| ADD <sup>4</sup>             | BWL             | s,Dn Dn,d               | *****                                   | e s s s s s s s s   | e d <sup>4</sup> d d d d d d     | s s s s s s s s   | s s s s s s s s   | s <sup>*</sup>  | s + Dn → Dn  | Dn + d → d  |  | Add binary (ADD) or ADDQ is used when source is #n. Prevent ADDQ with #n,L |             |                                       |  |
| ADDI <sup>4</sup>            | BWL             | #n,d                    | *****                                   | d - d d d d d d d   | d d d d d d d d                  | d d d d d d d d   | d d d d d d d d   | -               | s #n + d → d   |   | Add immediate to destination                             |  |             |                                       |  |
| ADDO <sup>4</sup>            | BWL             | #n,d                    | *****                                   | d d d d d d d d   | d d d d d d d d                  | d d d d d d 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 -(Ay),-(Ax)       | *****                                   | e - - - - - - - -   | e - - - - - - - -                | - - - - - - - -   | - - - - - - - -   | - - - - - - - - | Dy + Dx + X → Dx -(Ay) + -(Ax) + X → -(Ax)   |   | Add source and eXtend bit to destination                 |  |             |                                       |  |
| AND <sup>4</sup>             | BWL             | s,Dn Dn,d               | -**00                                   | e - s s s s s s s s   | e - d d d d d d d d              | s s s s s s s s   | s s s s s s s s   | s <sup>*</sup>  | s AND Dn → Dn  | Dn AND d → d  |  | Logical AND source to destination (ANDI is used when source is #n)         |             |                                       |  |
| ANDI <sup>4</sup>            | BWL             | #n,d                    | -**00                                   | d - d d d d d d d   | d d d d d d d d                  | d d d d d d d d   | d d d d d d d d   | -               | s #n AND d → d   |   | Logical AND immediate to destination                     |  |             |                                       |  |
| ANDI <sup>4</sup>            | B               | #n,CCR                  | =====                                   | - - - - - - - -   | - - - - - - - -                  | - - - - - - - -   | - - - - - - - -   | -               | s #n AND CCR → CCR   |   | Logical AND immediate to CCR                             |  |             |                                       |  |
| ANDI <sup>4</sup>            | W               | #n,SR                   | =====                                   | - - - - - - - -   | - - - - - - - -                  | - - - - - - - -   | - - - - - - - -   | -               | s #n AND SR → SR   |   | Logical AND immediate to SR (Privileged)                 |  |             |                                       |  |
| ASL                          | BWL             | Dx,Dy #n,Dy             | *****                                   | e - - - - - - - -   | d - - - - - - - -                | - - - - - - - -   | - - - - - - - -   | -               | x c l - - - - - - - -  | c - - - - - - - - x   |  | Arithmetic shift Dy by Dx bits left/right                                  |             |                                       |  |
| ASR                          | W               | d                       | *****                                   | d - - - - - - - -   | d d d d d d d d                  | d d d d d d d d   | d d d d d d d d   | -               | c - - - - - - - - x  | x c l - - - - - - - -   |  | Arithmetic shift Dy #n bits L/R (#n: 1 to 8)                               |             |                                       |  |
| Bcc                          | BW              | address <sup>2</sup>    | -----                                   | - - - - - - - -   | - - - - - - - -                  | - - - - - - - -   | - - - - - - - -   | -               | if cc true then address → PC   |   |  | Arithmetic shift ds 1 bit left/right (W only)                              |             |                                       |  |
| BCHG                         | B L             | Dn,d #n,d               | ---*--                                  | e <sup>1</sup> - d d d d d d d d                                    | d <sup>1</sup> - d d d d d d d d | d d d d d d d d   | d d d d d d d d   | -               | s NOT(bit number of d) → Z   |   |  | Branch conditionally (cc table on back) (B or 16-bit ± offset to address)  |             |                                       |  |
| BCLR                         | B L             | Dn,d #n,d               | ---*--                                  | e <sup>1</sup> - d d d d d d d d                                    | d <sup>1</sup> - d d d d d d d d | d d d d d d d d   | d d d d d d d d   | -               | s NOT(bit n of d) → bit n of d   |   |  | Set Z with state of specified bit in d then invert the bit in d            |             |                                       |  |
| BRA                          | BW <sup>3</sup> | address <sup>2</sup>    | ----                                    | - - - - - - - -   | - - - - - - - -                  | - - - - - - - -   | - - - - - - - -   | -               | s D → bit number of d  |   |  | Set Z with state of specified bit in d then clear the bit in d             |             |                                       |  |
| BSET                         | B L             | Dn,d #n,d               | ---*--                                  | e <sup>1</sup> - d d d d d d d d                                    | d <sup>1</sup> - d d d d d d d d | d d d d d d d d   | d d d d d d d d   | -               | s NOT(bit n of d) → Z  |   |  | Set Z with state of specified bit in d then set the bit in d               |             |                                       |  |
| BSR                          | BW <sup>3</sup> | address <sup>2</sup>    | ----                                    | - - - - - - - -   | - - - - - - - -                  | - - - - - - - -   | - - - - - - - -   | -               | PC → -(SP); address → PC   |   |  | Branch to subroutine (8 or 16-bit ± offset)                                |             |                                       |  |
| BTST                         | B L             | Dn,d #n,d               | ---*--                                  | e <sup>1</sup> - d d d d d d d d                                    | d <sup>1</sup> - d d d d d d d d | d d d d d d d d   | d d d d d d d d   | -               | s NOT(bit Dn of d) → Z   |   |  | Set Z with state of specified bit in d                                     |             |                                       |  |
| CHK                          | W               | s,Dn                    | -*UUU                                   | e - s s s s s s s s   | d - d d d d d d d d              | d d d d d d d d   | d d d d d d d d   | s               | s if Dn < 0 or Dn > s then TRAP  |   |  | Leave the bit in d unchanged   |             |                                       |  |
| CLR                          | BWL             | d                       | -0100                                   | d - d d d d d d d   | d - d d d d d d d                | d d d d d d d d   | d d d d d d d d   | -               | - d → d  |   |  | Compare Dn with 0 and upper bound [s]                                      |             |                                       |  |
| CMP <sup>4</sup>             | BWL             | s,Dn                    | *****                                   | e s <sup>4</sup> s s s s s s s s                                    | d s <sup>4</sup> s s s s s s s s | s s s s s s s s   | s s s s s s s s   | s <sup>*</sup>  | s set CCR with Dn - s  |   |  | Clear destination to zero  |             |                                       |  |
| CMPA <sup>4</sup>            | WL              | s,An                    | *****                                   | s e s s s s s s s s   | s d s s s s s s s s              | s s s s s s s s   | s s s s s s s s   | s               | s set CCR with An - s  |   |  | Compare Dn to source   |             |                                       |  |
| CMPI <sup>4</sup>            | BWL             | #n,d                    | *****                                   | d - d d d d d d d   | d d d d d d d d                  | d d d d d d d d   | d d d d d d d d   | -               | s set CCR with d - #n  |   |  | Compare An to source   |             |                                       |  |
| CMPM <sup>4</sup>            | BWL             | (Ay)+(Ax)+              | *****                                   | - - - e - - - -   | - - - d - - - -                  | - - - d - - - -   | - - - d - - - -   | -               | s set CCR with (Ax) - (Ay)   |   |  | Compare destination to #n  |             |                                       |  |
| DBcc                         | W               | Dn,address <sup>2</sup> | -----                                   | - - - - - - - -   | - - - - - - - -                  | - - - - - - - -   | - - - - - - - -   | -               | if cc false then { Dn-1 → Dn if Dn ↔ -1 then addn → PC } (16-bit ± offset to address)          |   |  | Test condition, decrement and branch (16-bit ± offset to address)          |             |                                       |  |
| DIVS                         | W               | s,Dn                    | -***0                                   | e - s s s s s s s s   | d - - - - - - - -                | d d d d d d d d   | d d d d d d d d   | s               | s ±32bit Dn / ±6bit s → ±Dn  |   |  | Dn = [ 16-bit remainder, 16-bit quotient ]                                 |             |                                       |  |
| DIVU                         | W               | s,Dn                    | -***0                                   | e - s s s s s s s s   | d - - - - - - - -                | d d d d d d d d   | d d d d d d d d   | s               | s 32bit Dn / 16bit s → Dn  |   |  | Dn = [ 16-bit remainder, 16-bit quotient ]                                 |             |                                       |  |
| EDR <sup>4</sup>             | BWL             | Dn,d                    | -**00                                   | e - d d d d d d d d   | d - - - - - - - -                | d d d d d d d d   | d d d d d d d d   | s <sup>*</sup>  | s Dn XOR d → d   |   |  | Logical exclusive OR Dn to destination                                     |             |                                       |  |
| EDRI <sup>4</sup>            | BWL             | #n,d                    | -**00                                   | d - d d d d d d d d   | d d d d d d d d                  | d d d d d d d d   | d d d d d d d d   | -               | s #n XOR d → d   |   |  | Logical exclusive OR #n to destination                                     |             |                                       |  |
| EDRI <sup>4</sup>            | B               | #n,CCR                  | =====                                   | - - - - - - - -   | - - - - - - - -                  | - - - - - - - -   | - - - - - - - -   | -               | s #n XOR CCR → CCR   |   |  | Logical exclusive OR #n to CCR   |             |                                       |  |
| EDRI <sup>4</sup>            | W               | #n,SR                   | =====                                   | - - - - - - - -   | - - - - - - - -                  | - - - - - - - -   | - - - - - - - -   | -               | s #n XOR SR → SR   |   |  | Logical exclusive OR #n to SR (Privileged)                                 |             |                                       |  |
| EXG                          | L               | Rx,Ry                   | -----                                   | e e - - - - - - - -   | d - - - - - - - -                | - - - - - - - -   | - - - - - - - -   | -               | s register ↔ register  |   |  | Exchange registers (32-bit only)   |             |                                       |  |
| EXT                          | WL              | Dn                      | -**00                                   | d - - - - - - - -   | 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 d d d d d d   | d 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 d d d d d d d   | d 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 s s s s s | - s s s s s s s s | s               | ↑s → An  |   |  | Load effective address of s to An  |             |                                       |  |
| LINK                         |                 | An,#n                   | -----                                   | - - - - - - - -   | - - - - - - - -                  | - - - - - - - -   | - - - - - - - -   | -               | An → -(SP); SP → An; SP + #n → SP  |   |  | Create local workspace on stack (negative n to allocate space)             |             |                                       |  |
| LSL                          | BWL             | Dx,Dy #n,Dy             | ***0*                                   | e - - - - - - - -   | d - - - - - - - -                | - - - - - - - -   | - - - - - - - -   | s <sup>*</sup>  | x c l - - - - - - - -  | c - - - - - - - - x   |  | Logical shift Dy, Dx bits left/right                                       |             |                                       |  |
| LSR                          | W               | d                       | ***0*                                   | - - - - - - - -   | - - - - - - - -                  | - - - - - - - -   | - - - - - - - -   | -               | c - - - - - - - - x  | x c l - - - - - - - -   |  | Logical shift Dy #n bits L/R (#n: 1 to 8)                                  |             |                                       |  |
| MOVE <sup>4</sup>            | BWL             | s,d                     | -**00                                   | e s <sup>4</sup> e e e e e e s s                                    | d - - - - - - - -                | d d d d d d d d   | d d d d d d d d   | s <sup>*</sup>  | s s → d  |   |  | Logical shift d 1 bit left/right (W only)                                  |             |                                       |  |
| MOVE                         | W               | s,CCR                   | =====                                   | s - s s s s s s s s   | s s s s s s s s                  | s s s s s s 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 s s s s s                  | s s s s s s 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   | d d d d d d d d                  | d d d d d d d d   | d d d d d d d d   | -               | SR → d   |   |  | Move Status Register to destination  |             |                                       |  |
| MOVE                         | L               | USP,An An,USP           | -----                                   | - d - - - - - - - -   | - s - - - - - - - -              | - - - - - - - -   | - - - - - - - -   | -               | USP → An   |   |  | Move User Stack Pointer to An (Privileged)                                 |             |                                       |  |
| MOVE                         |                 | An,USP                  | -----                                   | - s - - - - - - - -   | - 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 <sup>1</sup> | 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 <sup>1</sup> | WL   | Rn-Rn,d<br>s,Rn-Rn     |   | - - d - d d d d d            |       |        |           |    | Registers → d   |  |  |  |  |  |  |   |           | Move specified registers to/from memory<br>(W source is sign-extended to L for Rn)     |
| MOVEP              | WL   | Dn,(i.An)<br>(i.An),Dn |   | s - - - - d - - - -          |       |        |           |    | Dn → (i.An)...(i+2.An)...(i+4.A)                                |  |  |  |  |  |  |   |           | Move Dn to/from alternate memory bytes<br>(Access only even or odd addresses)          |
| MOVED <sup>1</sup> | 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 ±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 16bit s * 16bit Dn → Dn                                       |  |  |  |  |  |  |   |           | Multiply unsigned 16-bit result: unsigned 32-bit                                       |
| NBCD               | B    | d                      | *U*U*                                   | d - d d d d d d d            |       |        |           |    | D - d → X → d   |  |  |  |  |  |  |   |           | Negate BCD with eXtend, BCD result   |
| NEG                | BWL  | d                      | *****                                   | d - d d d d d d d            |       |        |           |    | D - d → d   |  |  |  |  |  |  |   |           | Negate destination (2's complement)  |
| NEGX               | BWL  | d                      | *****                                   | d - d d d d d d d            |       |        |           |    | D - d - X → d   |  |  |  |  |  |  |   |           | Negate destination with eXtend   |
| NDP                |      |                        |   | - - - - - - - -              |       |        |           |    |   |  |  |  |  |  |  |   |           | 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 <sup>4</sup>    | BWL  | s,Dn<br>Dn,d           | -**00                                   | e - s s s s s s s s          |       |        |           |    | s OR Dn → Dn  |  |  |  |  |  |  |   |           | Logical OR   |
| ORI <sup>4</sup>   | BWL  | #n,d                   | -**00                                   | d - d d d d d d d            |       |        |           |    | Dn OR d → d   |  |  |  |  |  |  |   |           | (ORi is used when source is #n)  |
| ORI <sup>4</sup>   | B    | #n,CCR                 |   | - - - - - - - -              |       |        |           |    | #n DR CCR → CCR   |  |  |  |  |  |  |   |           | Logical OR #n to CCR   |
| ORI <sup>4</sup>   | W    | #n,SR                  |   | - - - - - - - -              |       |        |           |    | #n DR SR → SR   |  |  |  |  |  |  |   |           | Logical OR #n to SR (Privileged)   |
| PEA                | L    | s                      |   | - - s - - - - s s s s        |       |        |           |    | Ts → (SP)   |  |  |  |  |  |  |   |           | Push effective address of s onto stack   |
| RESET              |      |                        |   | - - - - - - - -              |       |        |           |    | Assert RESET Line   |  |  |  |  |  |  |   |           | Issue a hardware RESET (Privileged)  |
| ROL                | BWL  | Dx,Dy<br>#n,Dy         | -**0*                                   | e - - - - - - - -            |       |        |           |    | C ← D <sub>Y</sub>  |  |  |  |  |  |  |   |           | Rotate Dy, Dx bits left/right (without X)  |
| RDR                | W    | d                      |   | d - - - - - - - -            |       |        |           |    | C ← D <sub>Y</sub>  |  |  |  |  |  |  |   |           | Rotate Dy, #n bits left/right (#n: 1 to 8)   |
| ROXL               | BWL  | Dx,Dy<br>#n,Dy         | ***0*                                   | e - - - - - - - -            |       |        |           |    | D - d → X → D <sub>Y</sub>                                      |  |  |  |  |  |  |   |           | Rotate Dy, Dx bits L/R, X used then updated  |
| ROXR               | W    | d                      |   | d - - - - - - - -            |       |        |           |    | D - d → X → D <sub>Y</sub>                                      |  |  |  |  |  |  |   |           | Rotate Dy, #n bits left/right (#n: 1 to 8)   |
|                    |      |                        |   |                              |       |        |           |    | D - d → X → D <sub>Y</sub>                                      |  |  |  |  |  |  |   |           | 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<br>(-Ay),(-Ax)   | *U*U*                                   | e - - - - - - - -            |       |        |           |    | D <sub>Y</sub> - D <sub>Y</sub> - X → D <sub>Y</sub>            |  |  |  |  |  |  |   |           | Subtract BCD source and eXtend bit from destination, BCD result                        |
| Scc                | B    | d                      |   | - - - - d - d d d d d d d    |       |        |           |    | (-Ax) <sub>Y</sub> - (Ay) <sub>Y</sub> - X → -(Ax) <sub>Y</sub> |  |  |  |  |  |  |   |           | If cc is true then 1's → d<br>else 0's → d<br>else dB = 11111111<br>else dB = 00000000 |
| STOP               |      | #n                     |   | - - - - - - - -              |       |        |           |    | #n → SR; STOP   |  |  |  |  |  |  |   |           | Move #n to SR, stop processor (Privileged)   |
| SUB <sup>4</sup>   | BWL  | s,Dn<br>Dn,d           | *****                                   | e - s s s s s s s s          |       |        |           |    | Dn - s → Dn   |  |  |  |  |  |  |   |           | Subtract binary (SUBI or SUBQ used when source is #n. Prevent SUBQ with #n,L)          |
| SUBA <sup>4</sup>  | WL   | s.An                   |   | e d <sup>4</sup> d d d d d d |       |        |           |    | d - Dn → d  |  |  |  |  |  |  |   |           | Subtract address (W sign-extended to L)  |
| SUBI <sup>4</sup>  | BWL  | #n,d                   | *****                                   | d - d d d d d d d            |       |        |           |    | s d - #n → d  |  |  |  |  |  |  |   |           | Subtract immediate from destination  |
| SUBQ <sup>4</sup>  | BWL  | #n,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<br>(-Ay),(-Ax)   | *****                                   | e - - - - - - - -            |       |        |           |    | Dx - Dy - X → Dx  |  |  |  |  |  |  |   |           | Subtract source and eXtend bit from 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   |                        |   | - - - - - - - -              |       |        |           |    | PC → (SSP); SR → (SSP);<br>(vector table entry) → PC            |  |  |  |  |  |  |   |           | Push PC and SR, PC set by vector table #n<br>(#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 (+ DR, ! NOT, ⊕ XOR; " Unsigned, " Alternate cc ) |                |          |    |                  |              |
|---|----------------|----------|----|------------------|--------------|
| cc  | Condition      | Test     | cc | Condition        | Test         |
| T   | true           | I        | Vc | overflow clear   | IV           |
| F   | false          | 0        | VS | overflow set     | V            |
| H <sup>10</sup>   | higher than    | I(C + Z) | PL | plus             | IN           |
| L <sup>8</sup>  | lower or same  | C + Z    | MN | minus            | N            |
| HS <sup>9</sup> , CC <sup>9</sup>                                 | higher or same | IC       | GE | greater or equal | I(N ⊕ V)     |
| LO <sup>9</sup> , CS <sup>9</sup>                                 | lower than     | C        | LT | less than        | (N ⊕ V)      |
| NE  | not equal      | IZ       | 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)  | SSP Supervisor Stack Pointer (32-bit)                 |
| Dn  | Data register (8/16/32-bit, n=0-7)   | USP User Stack Pointer (32-bit)                       |
| Rn  | any data or address register   | SP Active Stack Pointer (same as A7)                  |
| s   | Source, & Destination  | PC Program Counter (24-bit)                           |
| e   | Either source or destination   | SR Status Register (16-bit)                           |
| #n  | Immediate data, i Displacement   | CCR Condition Code Register (lower 8-bits of SR)      |
| BCD | Binary Coded Decimal   | N negative, Z zero, V overflow, C carry, X extend     |
| ↑   | Effective address  | * set according to operation's result, = set directly |
| 1   | Long only; all others are byte only  | - not affected, 0 cleared, 1 set, U undefined         |
| 2   | Assembler calculates offset  |   |
| 3   | Branch sizes: B or S -28 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

Distributed under the GNU general public use license.