

ALGO
QCM

1. La fonction d'essais successifs n'est pas utilisée par ?
 - (a) les méthodes indirectes de gestion des collisions
 - (b) le hachage avec Chaînage séparé
 - (c) le hachage coalescent

2. La méthode de hachage qui tronçonne la séquence de bits en sous-mots est ?
 - (a) la complétion
 - (b) la compression
 - (c) l'extraction
 - (d) la multiplication

3. Une fonction de hachage doit être déterministe ?
 - (a) Non
 - (b) Oui
 - (c) Cela dépend

4. Le handicap majeur de l'extraction est ?
 - (a) de hacher les anagrammes d'une clé de la même façon
 - (b) de nécessiter un m premier majorant le nombre de clés
 - (c) de n'utiliser qu'une partie de représentation de la clé
 - (d) de n'être efficace que sur une petite collection de données

5. Parmi les méthodes suivantes, lesquelles sont des méthodes de hachage de base ?
 - (a) division
 - (b) extraction
 - (c) compression
 - (d) multiplication

6. l'efficacité de la multiplication dépend ?
 - (a) principalement de m
 - (b) principalement de θ
 - (c) autant de m que de θ
 - (d) ni de m ni de θ

7. Quelles méthodes sont des méthodes indirectes de gestion des collisions ?
- (a) le hachage linéaire
 - (b) le double hachage
 - (c) le hachage coalescent
 - (d) le hachage avec chaînage séparé
8. Une collision secondaire représente une collision ?
- (a) avec coïncidence de valeur de hachage entre un x égal à un y
 - (b) sans coïncidence de valeur de hachage entre un x égal à un y
 - (c) sans coïncidence de valeur de hachage entre un x différent d'un y
 - (d) avec coïncidence de valeur de hachage entre un x différent d'un y
9. Le double hachage peut générer des collisions secondaires ?
- (a) Oui
 - (b) Non
 - (c) quelquefois
10. Quelles méthodes de hachage utilisent tous les bits de la représentation de la clé ?
- (a) la complétion
 - (b) la compression
 - (c) l'extraction
 - (d) la division



QCM N°3

lundi 10 octobre 2016

Question 11

Soit (u_n) une suite réelle convergente quelconque. Alors

- a. $\sum u_n$ converge
- b. $\sum(u_n - u_{n-1})$ converge
- c. $\sum(u_n - u_{n-1})$ diverge
- d. $\sum u_n$ converge absolument
- e. rien de ce qui précède

Question 12

Soit (u_n) une suite réelle positive, décroissante et convergeant vers 0. Alors

- a. $\sum(-1)^n u_n$ converge
- b. $\sum(-1)^n u_n$ diverge
- c. on ne peut rien dire sur la nature de $\sum(-1)^n u_n$

Question 13

- a. $\sum \frac{(-1)^n}{n}$ converge
- b. $\sum \frac{(-1)^n}{n}$ converge absolument
- c. $\sum \frac{1}{n}$ converge
- d. rien de ce qui précède

Question 14

Soit (u_n) une suite réelle telle que $\sum u_n$ converge absolument. Alors $\sum u_n$ converge.

- a. vrai
- b. faux

Question 15

Soit (u_n) une suite réelle telle que $u_n \underset{+\infty}{\sim} \frac{(-1)^n}{n}$. Alors

- a. $\sum u_n$ converge
- b. $\sum u_n$ diverge
- c. on ne peut rien dire sur la nature de $\sum u_n$

Question 16

Soit (u_n) une suite réelle quelconque telle que $\sum (u_n - u_{n-1})$ diverge. Alors

- a. $\sum u_n$ diverge
- b. (u_n) diverge
- c. (u_n) converge
- d. rien de ce qui précède

Question 17

Soient (u_n) et (v_n) deux suites réelles positives quelconques telles que $u_n = o(v_n)$ et $\sum v_n$ diverge. Alors

- a. $\sum u_n$ converge
- b. $v_n \rightarrow +\infty$
- c. $\sum u_n$ diverge
- d. on ne peut rien dire de la nature de $\sum u_n$

Question 18

Soit (u_n) une suite réelle strictement positive telle que

$$\frac{u_{n+1}}{u_n} \rightarrow \frac{1}{4}$$

Alors

- a. $\sum u_n$ converge
- b. $\sum u_n$ diverge
- c. on ne peut rien dire de la nature de $\sum u_n$

Question 19

Soit $\sum u_n$ une série à termes positifs et $(S_n) = \left(\sum_{k=1}^n u_k \right)$. Alors

- a. (S_n) est croissante
- b. (S_n) est décroissante
- c. (S_n) n'est pas nécessairement monotone
- d. $\sum u_n$ converge ssi (S_n) est majorée
- e. rien de ce qui précède

Question 20

Soit $\alpha \in \mathbb{R}$. Alors $\sum n^\alpha$

- a. converge ssi $\alpha > 1$
- b. converge ssi $\alpha < 1$
- c. converge ssi $\alpha < -1$
- d. converge ssi $\alpha > -1$
- e. diverge pour tout α

21. The sun ___ in the west every evening.
- a. set
 - b. should set
 - c. sets
 - d. will set
22. What did you do last night?
- a. I have eaten dinner.
 - b. I wrote some email and checked Facebook.
 - c. I was watching TV.
 - d. I had gone out with some friends.
23. Because of the force of gravity, objects...
- a. fall down and not up.
 - b. are falling down.
 - c. falls down.
 - d. are going to fall down.

A teacher is describing his students' actions from the front of the class (nos. 24 – 30). Choose the logical answer in each case.

24. "Yoko..."
- a. writes in her book."
 - b. has written in her book."
 - c. is writing in her book."
 - d. write in her book."
25. "Bill...."
- a. scratch his head."
 - b. is scratching his head."
 - c. will scratch his head."
 - d. Scratches his head."
26. "Dariush is staring out the window. He"
- a. seem daydreaming...
 - b. seem to be daydreaming...
 - c. is seeming to be daydreaming...
 - d. seems to be daydreaming...
27. "... but perhaps he"
- a. thinks hard about verb tenses."
 - b. is thinking hard about verb tenses."
 - c. has thought hard about verb tenses."
 - d. was thinking hard about verb tenses."
28. "What ___ Dariush ___?"
- a. you think / is doing
 - b. do you think / does
 - c. are you thinking / is doing
 - d. do you think / is doing
29. "Right now I ___ Nicole."
- a. I look
 - b. I look at
 - c. I'm looking at
 - d. I look to
30. "Nicole ___ angry."
- a. look
 - b. looks
 - c. is looking
 - d. looks like

31. Tylor's definition of culture includes:
- a. arts
 - b. morals
 - c. religious belief
 - d. all of the above
32. 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
33. The person responsible for bringing German philosophical perspectives to American anthropological theories on culture was:
- a. Wilhelm von Humboldt
 - b. Franz Boas
 - c. Hegel
 - d. Edward B. Tylor
34. The article "Shakespeare in the Bush" retells the story of which play:
- a. The Merchant of Venice
 - b. Macbeth
 - c. Julius Caesar
 - d. Hamlet
35. The Tiv are an ethnic group living in:
- a. West Africa
 - b. North America
 - c. South America
 - d. East Asia
36. Bohannon hopes that her telling Shakespeare's story to the Tiv will prove that it has:
- a. universal meaning
 - b. different meanings depending on the listener
 - c. different meanings depending on the language used
 - d. none of the above
37. What is an aspect of the story to which the Tiv react exactly as Bohannon expects?
- a. the appearance of the dead father's ghost
 - b. the fact that the chief's brother married his widow
 - c. both a and b
 - d. none of the above
38. The Tiv interpret the story according to:
- a. How Shakespeare intended it to be understood
 - b. universal values about family relationships and political power
 - c. their own cultural values and social organization
 - d. all of the above
39. "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

40. A cultural relativist would argue that all cultures are inherently _____, even if they differ from one another.
- a. the same
 - b. unable to be understood from the outside
 - c. equal
 - d. boring

Q.C.M n°3 de Physique

41- La force électrique qui décrit l'interaction entre deux charges ponctuelles q_1 et q_2 , séparées par une distance r est

- a) proportionnelle au produit des masses m_1 et m_2 des deux charges.
- b) prépondérante à l'échelle atomique
- c) inversement proportionnelle au produit des charges
- d) inversement proportionnelle à r

42- Un champ électrostatique \vec{E} est dit convergent lorsqu'il est créé par :

- a) Un proton
- b) Un neutron
- c) Un électron

43- Le champ électrostatique \vec{E} créé au point M par une charge placée au même point M est :

- a) convergent
- b) Nul
- c) divergent
- d) Non défini

44- Le champ électrostatique \vec{E} créé à l'infini par une charge placée au point O est :

- a) convergent
- b) Nul
- c) Non défini

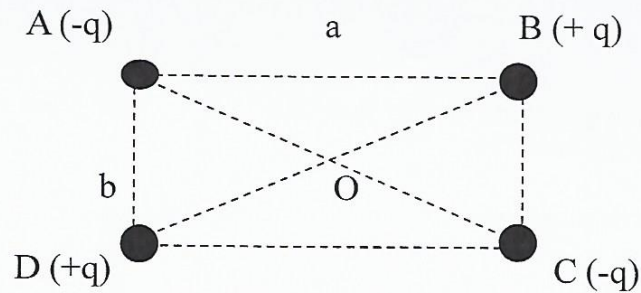
45- L'intensité du champ électrostatique créé au point M, par une charge q_A placée au point A est donné par :

a) $E_A(M) = k \frac{|q_A|}{(AM)^2}$ b) $E_A(M) = k \frac{|q_A| \cdot |q_M|}{(AM)^2}$ c) $E_A(M) = k \frac{|q_A|}{AM}$

46- Un doublet électrique $(-Q, +Q)$ de charges placées respectivement aux points A et B crée un champ électrique au milieu O du segment AB de norme :

a) $E(O) = k \frac{Q}{(AB)^2}$ b) $E(O) = \frac{4kQ}{(AB)^2}$ c) $E(O) = \frac{8kQ}{(AB)^2}$ d) $E(O) = 0$

47- On considère la distribution de charges suivante :



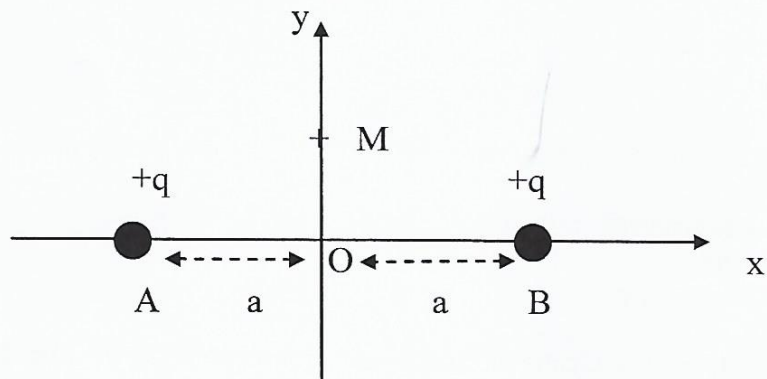
Le champ électrique créé au point O : centre du rectangle est

- a) orienté vers le point B b) infini **c) nul** d) orienté vers le point D

48- Dans le schéma ci-dessus, la force électrique exercée sur une charge (+q) que l'on place au centre O du rectangle est

- a) nulle** b) orientée vers le point B c) orientée vers le point D

49- On considère la distribution de charges suivante :



Le champ électrique créé au point O est

- a) nul** b) orienté vers le point B c) orienté vers le point A

50- Le champ électrique créé au point M, par la distribution de charges représentée ci-dessus (question 49) est

- a) parallèle à l'axe (Ox) b) nul **c) porté par l'axe (Oy)**

QCM Electronique – InfoS3

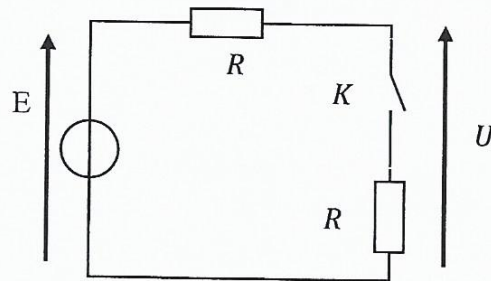
Pensez à bien lire les questions ET les réponses proposées (attention à la numérotation des réponses)

Révisions : Lois et Théorèmes de l'électronique

Q51. Soit le circuit ci-contre:

Quelle est la valeur de la tension U lorsque l'interrupteur K est ouvert?

- a- $U = 0$
- b- $U = \frac{E}{2}$
- c- $U = E$
- d- $U = -E$



Les semi-conducteurs et les diodes

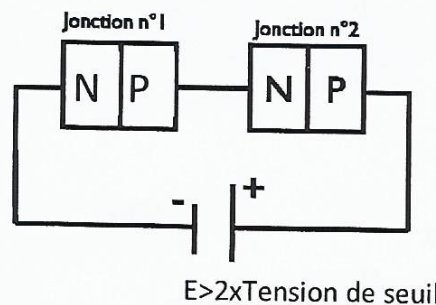
Q52. Le dopage permet d'augmenter la conductivité du semi-conducteur

- a- VRAI
- b- FAUX

Q53. Un semiconducteur intrinsèque est

- a- Un cristal pur.
- b- Un cristal dopé avec des atomes pentavalents
- c- Un cristal dopé avec des atomes trivalents
- d- Un cristal désordonné.

Q54.



Ce circuit est :

- a- Passant
- b- Bloqué

M

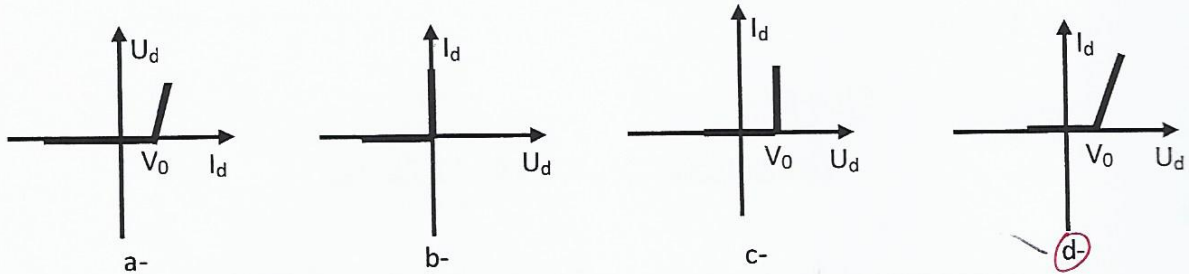
Q55. L'équation de la caractéristique de la diode s'écrit : $I_D = I_S(e^{\frac{V_D}{mV_T}} - 1)$ où I_D représente le courant qui traverse la diode et V_D , la tension à ses bornes, courant et tension étant fléchés selon la convention récepteur. I_S correspond au courant inverse. C'est un courant :

- a- Très grand (plusieurs dizaine 'ampères)
- b- Très faible (quelques nano ampères)

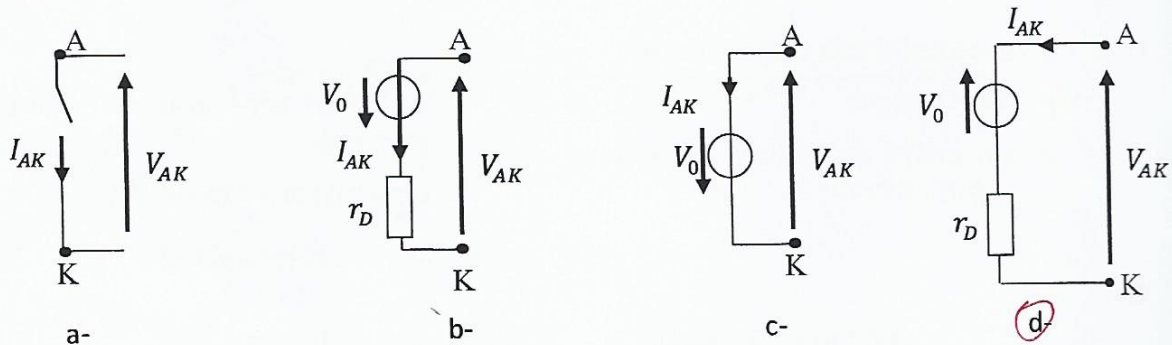
Q56. Quel modèle permet la représentation la moins précise de la diode :

- a- Le modèle idéal
- b- Le modèle à seuil
- c- Le modèle réel
- d- Les trois modèles sont équivalents

Q57. Laquelle de ces caractéristiques correspond à la caractéristique courant/tension du modèle réel de la diode :

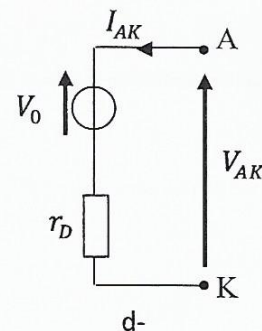
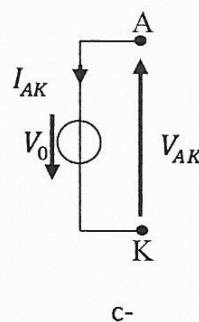
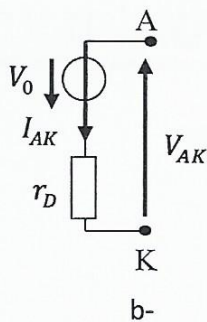
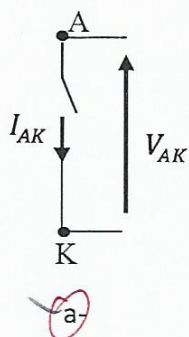


Q58. Par quoi remplace-t-on la diode passante si on utilise le modèle réel?



12

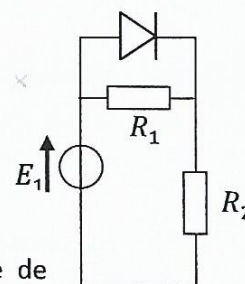
Q59. Par quoi remplace-t-on la diode bloquée si on utilise le modèle à seuil?



Q60. Soit le circuit ci-contre, dans lequel on considère la diode est telle que $V_0 = 0,6V$:

Choisir l'affirmation correcte si $E_1 = 10V$, $R_1 = 50\Omega$, et $R_2 = 1k\Omega$:

- (a-) La diode est bloquée et la tension à ses bornes est de l'ordre de $0,5V$. \times
- b- La diode est passante et le courant qui la traverse est de l'ordre de 10 mA
- c- La diode est passante et le courant qui la traverse vaut $-5A$.
- d- La diode est passante et le courant qui la traverse est de l'ordre de $9,4\text{ mA}$.



QCM 3

Architecture des ordinateurs

Lundi 10 octobre 2016

61. Quel mnémonique est une directive d'assemblage ?

- A. ORG
- B. MOVE
- C. ILLEGAL
- D. ADD

62. 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.

63. Soit l'instruction suivante : `MOVE.W 2(A0),D0`

- A. A0 est incrémenté de 1.
- B. A0 est incrémenté de 2.
- C. A0 est incrémenté de 4.
- D. A0 ne change pas.

64. Le registre CCR est : (deux réponses)

- A. Sur 8 bits.
- B. Sur 16 bits.
- C. Les 8 bits de poids fort du registre SR.
- D. Les 8 bits de poids faible du registre SR.

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

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

66. L'instruction BMI effectue un branchement si :

- A. $N = 0$
- B. $Z = 1$
- C. $N = 1$
- D. $Z = 0$

67. L'instruction BNE effectue un branchement si :

- A. $N = 0$
- ~~B. $Z = 1$~~
- C. $N = 1$
- D. $Z = 0$

68. Soient les deux instructions suivantes :

TST.B D0
BMI NEXT

L'instruction BMI effectue le branchement si :

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

69. 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. $D1 > D2$ (comparaison non signée)
- D. $D2 > D1$ (comparaison signée)

70. Soient les deux instructions suivantes :

CMP.L D1,D2
BLO NEXT

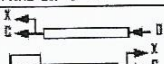
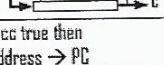
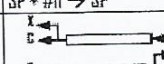
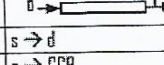
L'instruction BLO effectue le branchement si :

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

EASy68K Quick Reference v1.8

<http://www.wowgwp.com/EASy68K.htm>

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 - - - - - - - - - - | - - - - - - - - - - | $Dy_{10} + Dx_{10} + X \rightarrow Dx_{10}$ $-(Ay)_{10} + -(Ax)_{10} + X \rightarrow -(Ax)_{10}$ | Add BCD source and eXtend bit to destination, BCD result |
| ADD ⁴ | BWL | s,Dn Dn,d | ***** | e s s s s s s s s s s s s s s s s e d ⁴ d d d d d d d d d d - - - - | $s + Dn \rightarrow Dn$ $Dn + d \rightarrow d$ | Add binary (ADDI or ADDQ is used when 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 s s s s s | $s + An \rightarrow An$ | Add address (W sign-extended to .L) | |
| ADDI ⁴ | BWL | #n,d | ***** | d - d d d d d d d d d d - - - - | $\#n + d \rightarrow d$ | Add immediate to destination | |
| ADDQ ⁴ | BWL | #n,d | ***** | d - d d d d d d d d d d - - - - | $\#n + d \rightarrow d$ | Add quick immediate (#n range: 1 to 8) | |
| ADDX | BWL | Dy,Dx -(Ay),-(Ax) | ***** | e - - - - - - - - - - | - - - - - - - - - - | $Dy + Dx + X \rightarrow Dx$ $-(Ay) + -(Ax) + X \rightarrow -(Ax)$ | Add source and eXtend bit to destination |
| AND ⁴ | BWL | s,Dn Dn,d | ***00 | e - s s s s s s s s s s s s s s s s e - d d d d d d d d d d - - - - | $s \text{ AND } Dn \rightarrow Dn$ $Dn \text{ AND } d \rightarrow d$ | Logical AND source to destination (ANDI is used when source is #n) | |
| ANDI ⁴ | BWL | #n,d | ***00 | d - d d d d d d d d d d - - - - | $\#n \text{ AND } d \rightarrow d$ | Logical AND immediate to destination | |
| ANDI ⁴ | B | #n,CCR | ***00 | - - - - - | - - - - - | $\#n \text{ AND } CCR \rightarrow CCR$ | Logical AND immediate to CCR |
| ANDI ⁴ | W | #n,SR | ***00 | - - - - - | - - - - - | $\#n \text{ AND } SR \rightarrow SR$ | Logical AND immediate to SR (Privileged) |
| ASL | BWL | Dx,Dy | ***** | e - - - - - | - - - - - |  | Arithmetic shift Dy by Dx bits left/right |
| ASR | W | #n,Dy | ***** | d - d d d d d d d d d d - - - - | - - - - - |  | Arithmetic shift d 1 bit left/right (W only) |
| Bcc | BW ^d | address ² | ***** | - - - - - | - - - - - | if cc true then address \rightarrow PC | Branch conditionally (cc table on back) (8 or 16-bit \pm offset to address) |
| BCHG | B L | Dn,d #n,d | ***0* | e ¹ - d d d d d d d d d d - - - - d ¹ - d d d d d d d d d d - - - - | $\text{NOT}(\text{bit number of } d) \rightarrow Z$ $\text{NOT}(\text{bit } n \text{ of } d) \rightarrow \text{bit } n \text{ of } d$ | Set Z with state of specified bit in d then invert the bit in d | |
| BCLR | B L | Dn,d #n,d | ***0* | e ¹ - d d d d d d d d d d - - - - d ¹ - d d d d d d d d d d - - - - | $\text{NOT}(\text{bit number of } d) \rightarrow Z$ $0 \rightarrow \text{bit number of } d$ | Set Z with state of specified bit in d then clear the bit in d | |
| BRA | BW ^d | address ² | ***** | - - - - - | - - - - - | address \rightarrow PC | Branch always (8 or 16-bit \pm offset to address) |
| BSET | B L | Dn,d #n,d | ***0* | e ¹ - d d d d d d d d d d - - - - d ¹ - d d d d d d d d d d - - - - | $\text{NOT}(\text{bit } n \text{ of } d) \rightarrow Z$ $1 \rightarrow \text{bit } n \text{ of } d$ | Set Z with state of specified bit in d then set the bit in d | |
| BSP | BW ^d | address ² | ***** | - - - - - | - - - - - | PC \rightarrow -(SP); address \rightarrow PC | Branch to subroutine (8 or 16-bit \pm offset) |
| BST | B L | Dn,d #n,d | ***0* | e ¹ - d d d d d d d d d d - - - - d ¹ - d d d d d d d d d d - - - - | $\text{NOT}(\text{bit } Dn \text{ of } d) \rightarrow Z$ $\text{NOT}(\text{bit } \#n \text{ of } d) \rightarrow Z$ | Set Z with state of specified bit in d Leave the bit in d unchanged | |
| CHK | W | s,Dn | ***UUU | e - s s s s s s 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 d d d - - - - | $0 \rightarrow d$ | Clear destination to zero | |
| CMP ⁴ | BWL | s,Dn | ***** | e s ⁴ s s s s s s s s s s s s s s s s | 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 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 d d d - - - - | set CCR with $d - \#n$ | Compare destination to #n | |
| CMPM ⁴ | BWL | (Ay)+,(Ax)+ | ***** | - - - - - | - - - - - | set CCR with $(Ax) - (Ay)$ | Compare (Ax) to (Ay); Increment Ax and Ay |
| DBcc | W | Dn,address ² | ***** | - - - - - | - - - - - | if cc false then { $Dn - 1 \rightarrow Dn$ if $Dn < -1$ then $\text{addr} \rightarrow \text{PC}$ } | Test condition, decrement and branch (16-bit \pm offset to address) |
| DIVS | W | s,Dn | ***0* | e - s s s s s s s s s s s s s s s s | $\pm 32\text{bit } Dn / \pm 16\text{bit } s \rightarrow \pm Dn$ | $Dn = [16\text{-bit remainder}, 16\text{-bit quotient}]$ | |
| DIVU | W | s,Dn | ***0* | e - s s s s s s s s s s s s s s s s | $32\text{bit } Dn / 16\text{bit } s \rightarrow Dn$ | $Dn = [16\text{-bit remainder}, 16\text{-bit quotient}]$ | |
| EOR ⁴ | BWL | Dn,d | ***00 | e - d d d d d d d d d d - - - - | $Dn \text{ XOR } d \rightarrow d$ | Logical exclusive OR Dn to destination | |
| EDRI ⁴ | BWL | #n,d | ***00 | d - d d d d d d d d d d - - - - | $\#n \text{ XOR } d \rightarrow d$ | Logical exclusive OR #n to destination | |
| EDRI ⁴ | B | #n,CCR | ***00 | - - - - - | - - - - - | $\#n \text{ XOR } CCR \rightarrow CCR$ | Logical exclusive OR #n to CCR |
| EDRI ⁴ | W | #n,SR | ***00 | - - - - - | - - - - - | $\#n \text{ XOR } SR \rightarrow SR$ | Logical exclusive OR #n to SR (Privileged) |
| EXG | L | Rx,Ry | ***** | e e - - - - - | - - - - - | register \leftrightarrow register | Exchange registers (32-bit only) |
| EXT | WL | Dn | ***00 | d - - - - - | - - - - - | $Dn.B \rightarrow Dn.W$ $Dn.W \rightarrow Dn.L$ | Sign extend (change B to W or W to L) |
| ILLEGAL | | | ***** | - - - - - | - - - - - | PC \rightarrow -(SSP); SR \rightarrow -(SSP) | Generate illegal instruction exception |
| JMP | | d | ***** | - - d - - - - | - - d d d d d d d d d d - - - - | $\uparrow d \rightarrow \text{PC}$ | Jump to effective address of destination |
| JSR | | d | ***** | - - d - - - - | - - d d d d d d d d d d - - - - | PC \rightarrow -(SP); $\uparrow d \rightarrow \text{PC}$ | push PC, jump to subroutine at address d |
| LEA | L | s,An | ***** | - e s - - - - | - s s s s s s s s s s - - - - | $\uparrow s \rightarrow An$ | Load effective address of s to An |
| LINK | | An,#n | ***** | - - - - - | - - - - - | $An \rightarrow$ -(SP); $SP \rightarrow An$; $SP + \#n \rightarrow SP$ | Create local workspace on stack (negative n to allocate space) |
| LSL | BWL | Dx,Dy #n,Dy | ***0* | e - - - - - d - - - - - | - - - - - - - - - - |  | Logical shift Dy, Dx bits left/right |
| LSR | W | #n,Dy | ***0* | - - d d d d d d d d d d - - - - | - - - - - |  | Logical shift Dy, #n bits L/R (#n: 1 to 8) |
| MOVE ⁴ | BWL | s,d | ***00 | e s ⁴ e e e e e e e e e s s s s s s s | $s \rightarrow d$ | Move data from source to destination | |
| MOVE | W | s,CCR | ***** | s - s s s s s s s s s s s s s s s s | $s \rightarrow CCR$ | 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 | $s \rightarrow SR$ | Move source to Status Register (Privileged) | |
| MOVE | W | SR,d | ***** | d - d d d d d d d d d d - - - - | $SR \rightarrow d$ | Move Status Register to destination | |
| MOVE | L | USP,An An,USP | ***** | - d - - - - - - s - - - - - | - - - - - - - - - - | $USP \rightarrow An$ $An \rightarrow USP$ | Move User Stack Pointer to An (Privileged) 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 | | | |

16

| Opcode | Size | Operand | CCR | Effective Address s=source, d=destination, e=either, i=displacement | | | | | | | | | | | | Operation | Description | | |
|--------------------|------|------------------------|-------|---|----|------|-------|-------|--------|-----------|-------|-------|--------|-----------|----|-----------|-----------------|--|---|
| | | | | Dn | An | (An) | (An)+ | -(An) | (i.An) | (i.An,Rn) | abs.W | abs.L | (i.PC) | (i.PC,Rn) | #n | | | | |
| MOVEA ⁺ | W | s,An | XNZVC | s | e | s | s | s | s | s | s | s | s | s | s | s | s | s → An | Move source to An (MOVE s,An use MOVEA) |
| MOVEM ⁺ | W | Rn-Rn,d s,Rn-Rn | | - | - | d | - | d | d | d | d | d | - | - | - | - | - | Registers → d s → Registers | Move specified registers to/from memory (W source is sign-extended to L for Rn) |
| MOVEP | W | 0n,(i.An) (i.An),Dn | | s | - | - | - | - | d | - | - | - | - | - | - | - | - | 0n → (i.An)...(i+2.An)...(i+4.A) (i.An) → 0n... (i+2.An)...(i+4.A) | Move 0n to/from alternate memory bytes (Access only even or odd addresses) |
| MOVEQ ⁺ | 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 | 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 | 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 | - | - | - | - | 0 - d ₁₀ - X → d | Negate BCD with eXtend, BCD result |
| NEG | BWL | d | ***** | 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 | 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 ⁺ | BWL | s,Dn Dn,d | ***00 | e | - | s | s | s | s | s | s | s | s | s | s | s | s | s OR Dn → Dn Dn OR d → d | Logical OR (ORI is used when source is #n) |
| ORI ⁺ | BWL | #n,d | ***00 | d | - | d | d | d | d | d | d | d | - | - | - | s | #n OR d → d | Logical OR #n to destination | |
| ORI ⁺ | B | #n,CCR | | - | - | - | - | - | - | - | - | - | - | - | - | s | #n OR CCR → CCR | Logical OR #n to CCR | |
| ORI ⁺ | 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 | s | s | ↑s → -(SP) | Push effective address of s onto stack |
| RESET | | | | - | - | - | - | - | - | - | - | - | - | - | - | - | - | Assert RESET Line | Issue a hardware RESET (Privileged) |
| RDL | BWL | Dx,Dy | ***0* | e | - | - | - | - | - | - | - | - | - | - | - | - | - | c ← c | Rotate Dy, Dx bits left/right (without X) |
| RDR | W | #n,Dy d | | d | - | - | - | - | - | - | - | - | - | - | - | - | - | c ← c | Rotate Dy, #n bits left/right (#n: 1 to 8) |
| RDXL | BWL | Dx,Dy | ***0* | e | - | - | - | - | - | - | - | - | - | - | - | - | - | c ← c | Rotate d 1-bit left/right (W only) |
| RDXR | W | #n,Dy d | | d | - | - | - | - | - | - | - | - | - | - | - | - | - | c ← c | Rotate Dy, Dx bits L/R, X used then updated |
| 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 -(Ay),-(Ax) | *U*U* | e | - | - | - | - | - | - | - | - | - | - | - | - | - | Dx ₁₀ - Dy ₁₀ - X → Dx ₁₀ -(Ax) ₁₀ - (Ay) ₁₀ - X → -(Ax) ₁₀ | Subtract BCD source and eXtend bit from destination, BCD result |
| SCC | B | d | | d | - | d | d | d | d | d | d | d | - | - | - | - | - | If cc is true then f'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 ⁺ | BWL | s,Dn Dn,d | ***** | e | s | s | s | s | s | s | s | s | s | s | s | s | s | Dn - s → Dn d - Dn → d | Subtract binary (SUBI or SUBQ used when source is #n. Prevent SUBQ with #n.L) |
| SUBA ⁺ | W | s,An | | s | e | s | s | s | s | s | s | s | s | s | s | s | s | An - s → An | Subtract address (W sign-extended to L) |
| SUBI ⁺ | BWL | #n,d | ***** | d | - | d | d | d | d | d | d | d | - | - | - | - | - | d - #n → d | Subtract immediate from destination |
| SUBQ ⁺ | 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 -(Ay),-(Ax) | ***** | e | - | - | - | - | - | - | - | - | - | - | - | - | - | Dx - Dy - X → Dx -(Ax) - (Ay) - X → -(Ax) | 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 | | - | - | - | - | - | - | - | - | - | - | - | - | - | 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 |

| cc | Condition | Test | cc | Condition | Test |
|-----------------------------------|----------------|----------|----|------------------|----------------|
| T | true | I | VC | overflow clear | IV |
| F | false | 0 | VS | overflow set | V |
| HI [°] | higher than | I(C + Z) | PL | plus | IN |
| LS [°] | lower or same | C + Z | MI | minus | N |
| HS [°] , CC [°] | higher or same | IC | GE | greater or equal | !(N ⊕ V) |
| LD [°] , BS [°] | 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)
 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, L, 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 Cszasz, Lawrence Tech University – 2004-2006

Distributed under the GNU general public use license.

17