# ALGO
# QCM

1. **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

2. **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

3. **Dans un graphe non orienté G=<S,A>, Le sous-graphe connexe maximal G'=<S',A> est une composante connexe du graphe G ?**
   (a) vrai
   (b) faux

4. **Un graphe partiel G' de G=<S,A> est défini par ?**
   (a) <S,A'> avec A' $\subseteq$ A
   (b) <S',A> avec S' $\subseteq$ S
   (c) <A,S>

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. **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

7. **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 inférieur à Pref[x] dans la forêt sont appelés ?**
   (a) Arcs couvrants
   (b) Arcs en arrière
   (c) Arcs en Avant
   (d) Arcs croisés

8. **Dans un graphe valué G=<S,A,C>, 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 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

QCM 6 Azar Chap20 (condits3 ex 16) fall 23 (late bus)

Choose the one correct answer for each sentence. One answer only unless otherwise indicated.

21. The bus you take to school has been late every day this week. You say:
a. If the bus had arrived on time, I would not have been late for class.
b. If the bus had arrive on time, I would not have been late for class.
c. If the bus arrived on time, I would not have been late for class.
d. If the bus arrived on time, I would not have be late for class.

22. The sentence, "If the team had practiced more, they would have won," refers to:
a. the past.
b. the future.
c. the present and the future.
d. the present.

23. You say: "If Kengo had written the paper by himself, there would have been many mistakes." In truth, this means:
a. Kengo wrote the paper by himself.
b. There were many mistakes.
c. Kengo did not write the paper by himself.
d. Kengo is angry about the paper.

24. If you say: "If my sister were rich, she would buy Tesla," this refers to:
a. The future.
b. The past.
c. The present.

25. If William ___ to class late today, I _____ him in.
a. came / would not have let
b. had come / will not
c. didn't come / would not
d. had come / would not have let

26. Hank tried to send the president a warning by email last night, but he didn't have enough time. In other words:
a. If Hank had enough time, he would have sent her a warning.
b. If Hank hadn't enough time, he would have sent her a warning.
c. If Hank had had enough time, he would have sent her a warning.
d. If Hank had had enough time, he would send her a warning.

27. Choose the **one** correct sentence.
a. If I had been born rich, I will not study.
b. If I had been born rich, I was happy.
c. If I had been born rich, I will be happy.
d. If I had been born rich, I would not have gone to school.

28. ____ not all the spectators had arrived, the match took place on time, as planned.

a. When

b. If

c. Even though

d. Whether

29. Caroline wants to change heaters because the one she has is old. Which sentence matches?

a. If her heater were newer, she would have kept it.

b. If her heater were newer, she would not think about changing it.

c. If her heater would be newer, she would keep it.

d. If her heater were newer, she keeps it.

30. Which TWO sentences are perfectly correct?

a. I would have bought the stock only if interest rates had gone down.

b. I will buy the stock only if interest rates goes down.

c. I will buy the stock only if interest rates go down.

d. I will have bought the stock only if interest rates go down.

## **QCM 6** – OC S3 2023/24 (Week 20 November)

**31.** Non-verbal communication can _____________. *Choose all that apply*

a) complement a verbal message.
b) accentuate verbal communication.
c) contradict a verbal message.
d) None of the above

**32.** A key difference between verbal and non-verbal communication is that ________

a) Verbal communication is nonlinear.
b) Non-verbal communication is linear.
c) Verbal communication is linear and nonverbal communication is nonlinear.
d) There are no specific differences between verbal and non-verbal cues.

**33.** When a teacher pauses during a lecture and looks at students who are talking in order to communicate that they should be quiet, what function is being fulfilled by the non-verbal message?

a) accenting
b) complementing
c) substituting
d) contradicting

**34.** Which of the following is **NOT** a characteristic of non-verbal communication?

a) It remains unaffected by its setting.
b) It often operates at a subconscious level.
c) It reveals feelings and attitudes.
d) It may conflict with verbal messages.

**35.** Which of the following statements best describes paralanguage?

a) It involves the speaker's choice of words.
b) It can create a distinct impression of the speaker.
c) Its main component is body language.
d) It exists beside language and interacts with it.

**36.** Which of the following is **NOT** an aspect of paralanguage?

a) facial expressions
b) rate of speech
c) pitch of voice
d) volume of voice

**37.** Displays of feelings can vary by culture. Which of the following is **NOT** true?

a) Smiling is generally considered a positive sign.
b) Many West African cultures tend to openly express emotions.
c) Americans only smile when they are happy.
d) In some cultures, excessive smiling may signal shallowness.

**38.** In all cultures, smiling a lot is seen as a good thing. True or False?

a) True
b) False

**39.** Non-verbal communication skills are something that we are born with and can't be learnt. True or False?

a) True
b) False

**40.** Which country according to the Preferred Interpersonal Distances 2017 study had the shortest personal space preference between themselves and a stranger?

a) Bulgaria
b) Romania
c) Argentina
d) Italy

# QCM Physique – InfoS3 – 20.11

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

Q41. Selon des mesures expérimentales, pour un objet réel ayant un comportement proche du corps noir, le spectre de son rayonnement (intensité du rayonnement en fonction de la longueur d'onde λ) dépend de la température de celui-ci.

a. Vrai
b. Faux

Q42. Le rayonnement du corps noir, décrit par la loi de Rayleigh-Jeans, est décrit comme la « catastrophe ultraviolette » car :

a. La densité d'énergie rayonnée diverge vers +∞ pour les courtes longueurs d'onde.
b. La densité d'énergie rayonnée diverge vers +∞ pour les grandes longueurs d'onde.
c. La densité d'énergie rayonnée pour les grandes longueurs d'onde est nulle
d. La densité d'énergie rayonnée pour les petites longueurs d'onde est nulle

Q43. Le quanta d'énergie a pour expression :

a. $E_0 = hc\lambda$
b. $E_0 = h\lambda$
c. $E_0 = \frac{hc}{\lambda}$
d. $E_0 = \frac{hc}{\lambda^2}$

Q44. Sur l'effet photoélectrique, on peut dire :

a. Qu'il correspond à l'émission d'un photon par irradiation d'un métal par un faisceau d'électrons.

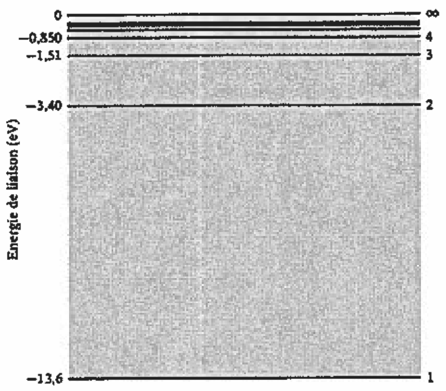
b. Qu'il correspond à l'émission d'un électron par irradiation d'un métal par un faisceau lumineux.

c. Qu'il n'a lieu qu'à partir d'une certaine énergie apportée.
d. Qu'il a lieu peu importe l'énergie apporté.

Q45. Selon le modèle de Bohr de l'atome d'hydrogène, lors de la désexcitation d'un électron d'un niveau supérieur vers un niveau inférieur, il y a :

a. Absorption d'un quanta d'énergie.
b. Emission d'un quanta d'énergie.

Les Q46&47. s'appuient sur le diagramme d'énergie ci-dessous de l'atome d'hydrogène de Bohr.



$E_1 = -13{,}6\ eV$

$E_2 = -3{,}4\ eV$

$E_3 = -1{,}51\ eV$

$E_4 = -0{,}850\ eV$

$E_5 = -0{,}54\ eV$

$E_6 = -0{,}37\ eV$

$E_\infty = 0$

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

Q46. L'énergie à fournir pour passer de l'état fondamental à l'orbite n = 3 est égale à :

a. 12,09 ev
b. -12,09 ev
c. 1,51 ev
d. -1,51 ev

Q47. La longueur d'onde correspondant à une transition de l'état n = 3 vers l'état n' = 2 vaut :

a. $\lambda = hc\,|\Delta E_{3\to2}|$
b. $\lambda = hc\,\Delta E_{3\to2}$
c. $\lambda = \frac{hc}{|\Delta E_{3\to2}|}$
d. $\lambda = \frac{hc}{\Delta E_{3\to2}}$

Q48. Selon le modèle de Bohr de l'atome d'hydrogène, le rayon d'une orbite électronique numérotée $n \in \mathbb{N}^*$est lié au rayon $a_0$ de l'orbite de plus basse énergie par la relation :

a. $r_n = a_0 n$ ; n = $1, 2, 3 \dots$
b. $r_n = a_0 n^2$ ; n = $1, 2, 3 \dots$
c. $r_n = \frac{a_0}{n^2}$ ; n = $1, 2, 3 \dots$
d. $r_n = \frac{a_0}{n}$ ; n = $1, 2, 3 \dots$

Q49. Selon le modèle de Bohr de l'atome d'hydrogène, l'énergie d'une orbite électronique numérotée $n \in \mathbb{N}^*$est lié à l'énergie $E_1$ de l'orbite de plus basse énergie par la relation :
a. $E_n = E_1 n^2; n = 1, 2, 3 \dots$
b. $E_n = E_1 n\,; n = 1, 2, 3 \dots$
c. $E_n = \frac{E_1}{n^2}\,; n = 1, 2, 3 \dots$
d. $E_n = \frac{E_1}{n}\,; n = 1, 2, 3 \dots$

Q50. Le spectre lumineux visible de l'hydrogène est :
a. Un continuum de longueurs d'onde du violet au rouge d'intensité constante.
b. Un continuum de longueurs d'onde du violet au rouge avec un pic d'intensité pour une certaine longueur d'onde.
c. Un spectre composé de plusieurs raies lumineuses distinctes.
d. Un spectre composé d'une seule raie lumineuse.

# QCM 6
# Architecture des ordinateurs

**Lundi 20 novembre 2023**

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

51. Choisir les réponses correctes.
    A. Un mot de 16 bits peut être empilé.
    B. Un mot de 32 bits peut être empilé.
    C. Un octet peut être empilé.
    D. Aucune de ces réponses.

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

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

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

55. Soient les deux instructions suivantes :

```
CMP.B D1,D2
BLE   NEXT
```

Branchement à NEXT si :

A. D1 = \$18929218 et D2 = \$92181892
B. D1 = \$92181892 et D2 = \$92181892
C. D1 = \$18929218 et D2 = \$18929218
D. D1 = \$92181892 et D2 = \$18929218

56. Quelle(s) instruction(s) n'est (ne sont) pas possible(s) ?

A. SUBQ.L #3,D0
B. SUBQ.L #42,D3
C. SUBQ.L #8,A2
D. SUBQ.B #2,(A2)

57. Quelle(s) instruction(s) n'est (ne sont) pas possible(s) ?

A. SUBI.L #42,D0
B. SUBI.L D2,D3
C. SUBI.L #8,A2
D. SUBI.B #2,(A2)

58. Quelle(s) instruction(s) n'est (ne sont) pas possible(s) ?

A. MOVEQ.L D1,D0
B. MOVEQ.B #42,D0
C. MOVEQ.W #42,D0
D. MOVEQ.L #42,D0

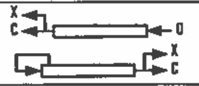
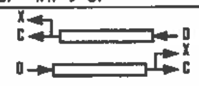
59. Quelle(s) instruction(s) n'est (ne sont) pas possible(s) ?

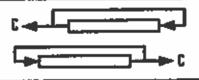
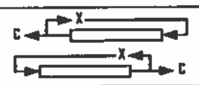
A. MOVEA.L #50,D0
B. MOVEA.L #50,A0
C. MOVEA.B #50,D0
D. MOVEA.B #50,A0

60. Quelle(s) instruction(s) n'est (ne sont) pas possible(s) ?

A. SWAP.W D7
B. SWAP.W A1
C. SWAP.L A4
D. SWAP.B D7

**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<br>s,Rn-Rn | ----- | -<br>- | -<br>- | d<br>s | -<br>s | d<br>- | d<br>s | d<br>s | d<br>s | d<br>s | -<br>s | -<br>s | -<br>- | Registers → d<br>s → Registers | 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<br>d | -<br>- | -<br>- | -<br>- | -<br>- | d<br>s | -<br>- | -<br>- | -<br>- | -<br>- | -<br>- | -<br>- | Dn → (i,An)...(i+2,An)...(i+4,A.<br>(i,An) → Dn...(i+2,An)...(i+4,A. | Move Dn to/from alternate memory bytes<br>(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 | - | - | - | $0 - d_{10} - X \rightarrow d$ | 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<br>Dn,d | -**00 | e<br>e | -<br>- | s<br>d | s<br>d | s<br>d | s<br>d | s<br>d | s<br>d | s<br>d | s<br>- | s<br>- | s$^4$<br>- | s OR Dn → Dn<br>Dn OR d → d | Logical OR<br>(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 |
| DRI $^4$ | B | #n,CCR | ===== | - | - | - | - | - | - | - | - | - | - | - | s | #n OR CCR → CCR | Logical OR #n to CCR |
| DRI $^4$ | 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<br>ROR | BWL<br><br>W | Dx,Dy<br>#n,Dy<br>d | -**0* | e<br>d<br>- | -<br>-<br>- | -<br>-<br>d | -<br>-<br>d | -<br>-<br>d | -<br>-<br>d | -<br>-<br>d | -<br>-<br>d | -<br>-<br>d | -<br>-<br>- | -<br>-<br>- | -<br>s<br>- |  | Rotate Dy, Dx bits left/right (without X)<br>Rotate Dy, #n bits left/right (#n: 1 to 8)<br>Rotate d 1-bit left/right (.W only) |
| ROXL<br>ROXR | BWL<br><br>W | Dx,Dy<br>#n,Dy<br>d | ***0* | e<br>d<br>- | -<br>-<br>- | -<br>-<br>d | -<br>-<br>d | -<br>-<br>d | -<br>-<br>d | -<br>-<br>d | -<br>-<br>d | -<br>-<br>d | -<br>-<br>- | -<br>-<br>- | -<br>s<br>- |  | Rotate Dy, Dx bits L/R, X used then updated<br>Rotate Dy, #n bits left/right (#n: 1 to 8)<br>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<br>- | -<br>- | -<br>- | -<br>- | -<br>e | -<br>- | -<br>- | -<br>- | -<br>- | -<br>- | -<br>- | -<br>- | $Dx_{10} - Dy_{10} - X \rightarrow Dx_{10}$<br>$-(Ax)_{10} - -(Ay)_{10} - X \rightarrow -(Ax)_{10}$ | Subtract BCD source and eXtend bit from<br>destination, BCD result |
| Scc | B | d | ----- | d | - | d | d | d | d | d | d | d | - | - | - | If cc is true then 1's → d<br>else 0's → d | If cc true then d.B = 11111111<br>else d.B = 00000000 |
| STOP | | #n | ===== | - | - | - | - | - | - | - | - | - | - | - | s | #n → SR; STOP | Move #n to SR, stop processor (Privileged) |
| SUB $^4$ | BWL | s,Dn<br>Dn,d | ***** | e<br>e | s<br>d$^4$ | s<br>d | s<br>d | s<br>d | s<br>d | s<br>d | s<br>d | s<br>d | s<br>- | s<br>- | s$^4$<br>- | Dn - s → Dn<br>d - Dn → d | Subtract binary (SUBI or SUBQ used when<br>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<br>-(Ay),-(Ax) | ***** | e<br>- | -<br>- | -<br>- | -<br>- | -<br>e | -<br>- | -<br>- | -<br>- | -<br>- | -<br>- | -<br>- | -<br>- | Dx - Dy - X → Dx<br>-(Ax) - -(Ay) – X → -(Ax) | Subtract source and eXtend bit from<br>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);<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 (+ OR, ! NOT, ⊕ XOR; $^u$ Unsigned, $^a$ Alternate cc ) | | | | | |
|---|---|---|---|---|---|
| cc | Condition | Test | cc | Condition | Test |
| T | true | 1 | VC | overflow clear | !V |
| F | false | 0 | VS | overflow set | V |
| HI$^u$ | higher than | !(C + Z) | PL | plus | !N |
| LS$^u$ | lower or same | C + Z | MI | minus | N |
| HS$^u$, CC$^a$ | higher or same | !C | GE | greater or equal | !(N ⊕ V) |
| LO$^u$, CS$^a$ | 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 |

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

| | |
|---|---|
| **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 |