



**UNIVERSITI KUALA LUMPUR
Malaysia France Institute**

**FINAL EXAMINATION
SEPTEMBER 2013 SESSION**

SUBJECT CODE : FSD23102
SUBJECT TITLE : MICROPROCESSOR
LEVEL : DIPLOMA
TIME / DURATION :
(2 HOURS)
DATE :

INSTRUCTIONS TO CANDIDATES

1. Please read the instructions given in the question paper **CAREFULLY**.
2. This question paper is printed on both sides of the paper.
3. Please write your answers on the answer booklet provided.
4. Answer should be written in blue or black ink except for sketching, graphic and illustration.
5. This question paper consists of **TWO (2)** sections. Section A and B. Answer all questions in Section A. For Section B, answer two (2) questions only.
6. Answer all questions in English.

THERE ARE 9 PAGES OF QUESTIONS AND 2 PAGES OF APPENDICES, EXCLUDING THIS PAGE.

SECTION A (Total: 60 marks)

**INSTRUCTION: Answer ALL questions.
Please use the answer booklet provided.**

Question 1

- (a) Provide the definition of microprocessor. (2 marks)
- (b) State TWO (2) system buses in M68K microprocessor. (2 marks)
- (c) Describe the principles of CPU Execution Cycle for M68K Microprocessor. (3 marks)
- (d) CPU is the “master” component in Microprocessor System. List THREE (3) main components of CPU. (3 marks)
- (e) Define Reprogrammable System and Embedded System and give TWO (2) examples for each application. (6 marks)
- (f) Differentiate between RAM and ROM in terms of data storage and computer application process performed by both memory types. (4 marks)

Question 2

(a) Fill in the blanks with correct answers.

i. The function of Condition Code Register is _____. This register consist of _____ bits flags which are _____.
(3 marks)

ii. 1 long word = _____ words = _____ bytes = _____ bits.
(3 marks)

iii. If the interrupt signal level has the highest priority the status for Interrupt Mask Bits (IMB) is _____.
(1 mark)

(b) Describe the function of Program Counter and state the size in bits of this register.
(3 marks)

(c) Figure 1 shows the Pin Assignment for M68K microprocessor. Based on the figure answer the following questions:

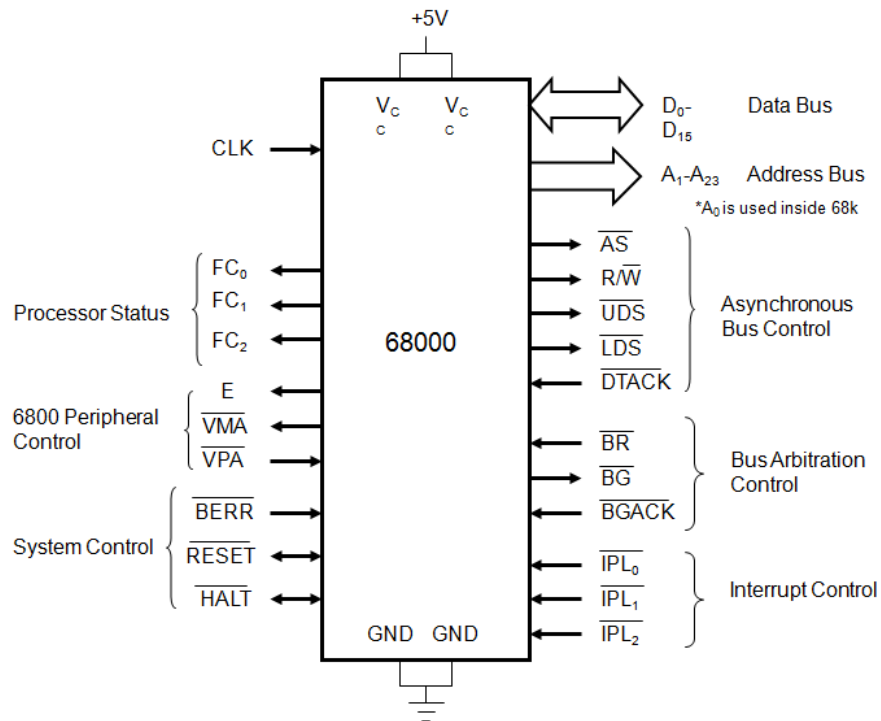


Figure 1: Pin Assignment for M68K microprocessor.

- i. List THREE (3) states of RW pin in Asynchronous Bus Control. (3 marks)
 - ii. Describe the function of System Control pins. (3 marks)
- (d) Figure 2 shows the example of Interrupt Process. The External Peripheral has important task while M68K is executing its instructions normally. The interrupt signal has been request by External Peripheral Interrupt Control Pin. If the external interrupt higher than current process, briefly explain what are the next steps performed by M68K.

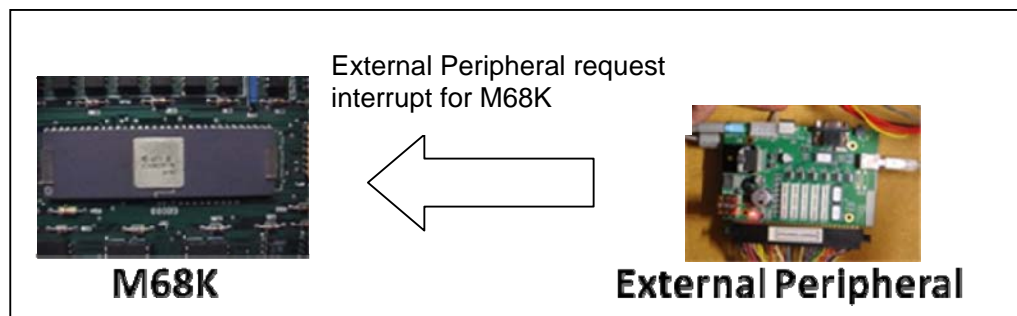


Figure 2: Interrupt Process

(4 marks)

Question 3

Convert and perform the arithmetic operation below and show the conversion procedure algorithmically.

(a) Convert **453** to hexadecimal form. (3 marks)

(b) Convert **\$F4A** to binary form. (2 marks)

(c) Convert **%10011.111** to decimal form. (2 marks)

(d) Convert signed number **\$FC** to decimal form. (3 marks)

(e) By using *two's complement* binary arithmetic, compute the following operation.

Note: *Your calculations should be in 8-bit format for integer numbers.*

$$70 - \$1F$$

(5 marks)

(f) Based on your answers in Question 3(d), state the status of C-bit and Z-bit in Condition Code Register. (2 marks)

(g) Describe the function of V-bit in Condition Code Register and state the status based on your answers in Question 3 (d). (3 marks)

SECTION B (Total: 40 marks)

INSTRUCTION: Answer TWO (2) questions only

Please use the answer booklet provided.

Question 4

- (a) Figure 3 shows the initial values of Address Registers, Data Registers and memory locations in M68K microprocessor.

Initial Values for Address & Data Registers	Initial Memory	
A1 = \$600601	\$600600	\$58
A2 = \$600605	\$600601	\$63
A3 = \$60060A	\$600602	\$24
D0 = \$22224444	\$600603	\$12
D1 = \$FEDCBA12	\$600604	\$00
D2 = \$55556666	\$600605	\$FF
D3 = \$00000007	\$600606	\$02
	\$600607	\$BB
	\$600608	\$00
	\$600609	\$00
	\$60060A	\$05

Figure 3: Initial values for Address Registers, Data Registers and memory.

Explain the contents of the affected registers or memory locations when each of the following instructions are executed. Each instruction is executed independently. The initial values of the registers and memory are the same before each instruction is executed.

- i. ADD .B (A3), D3 (2 marks)
- ii. MOVE .B \$06(A1), D0 (2 marks)
- iii. MOVE . L D2, \$600600 (2 marks)
- iv. MOVE . B \$600605, D1 (2 marks)

- (b) Find the destination target and source for instruction code below.

MOVE.W D1, D0

(2 marks)

- (c) Write complete assembly language programs to divide unsigned numbers \$4D with \$04. Your programs also should store \$4D and \$04 in two data registers and start with memory location \$1000.

(5 marks)

- (d) Based on the Question 4(c), compute the answers from that division operation and illustrate it in terms of data arrangement. State the status for V-bit flag based on the final answer.

(5 marks)

Question 5

- (a) Table 1 shows four (4) unsigned 8-bits data that need to be stored into allocated memory address. Then, these four (4) data will be added and the result will be stored in D0. Consider the assembly language programs below and continue the programs with your answers by using do-while loop. Your programs should be complete with comments.

Table 1: Data stored in equivalent memory address

Data Value	Memory Address
#\$2A	\$3000
#\$41	\$3001
#\$43	\$3002
#\$30	\$3003

```

START      ORG      $400400
           MOVE.B   #$2A,$3000 ; Put $2A into address $3000
           MOVE.B   #$41,$3001 ; Put $41 into address $3001
           MOVE.B   #$43,$3002 ; Put $43 into address $3002
           MOVE.B   #$30,$3003 ; Put $30 into address $3002
    
```

(15 marks)

- (b) Consider the assembly language instruction below and discuss on the Addressing Mode for this instruction.

```

MOVE.B     #$2A,$3000
    
```

(5 marks)

Question 6

- (a) A post service company needs a system to calculate the volume of rectangular box for international shipping services. As a software engineer, you need to create an assembly language programs that will calculate the volume of rectangular box for this post service company based on the following information:
- i. Memory locations \$400000 until \$400002 contain the width, height and length of rectangular, respectively as in Table 2.

Table 2: Data stored in equivalent Memory Address

Memory Address	Data Value
\$400000	#\$03
\$400001	#\$02
\$400002	#\$05

- ii. Your programs should consist of **Address Register Indirect Addressing Mode**.
- iii. Use this formula given for calculating the volume of rectangle:

$$\text{Volume of rectangle} = \text{width} \times \text{height} \times \text{length}$$
- iv. The result must be stored at memory address \$700002.

Consider the assembly language programs below and please continue the programs with your answers based on the above informations. Your programs should be complete with comments.

```

START    ORG    $1000; programs origin start at memory $1000
         LEA    $400000,A0; Load affected address $400000 into A0
         LEA    $400001,A1; Load affected address $400001 into A1
         LEA    $400002,A2; Load affected address $400002 into A2
    
```

(15 marks)

(b) Based on Question 6 (a) compute the result stored in memory address \$700002.
Prove your answer with calculation.


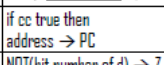
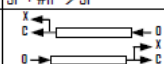
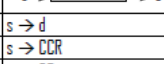
(3 marks)

(c) Briefly explain on the MOVEA instruction code.

(2 marks)

END OF QUESTIONS

APPENDIX 1: M68K Datasheet

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) (iAn) (iAn,Rn) abs.W abs.L (iPC) (iPC,Rn) #n			
ABCD	B	Dy,Dx -(Ay)-:(Ax)	*U*U*	e - - - - - d' - - - - -	- - - - - - - - - -	$Dy_0 + Dx_{10} + X \rightarrow Dx_{10}$ $-(Ay)_0 + -(Ax)_0 + X \rightarrow -(Ax)_0$	Add BCD source and eXtend bit to destination, BCD result
ADD ⁺	BWL	s,Dn Dn,d	*****	e s s s s s s s s s s s ⁺ e d' d d d d d d d d d - -	- - - - - - - - - -	$s + Dn \rightarrow Dn$ $Dn + d \rightarrow d$	Add binary (ADD) or ADDQ is used when source is #n. Prevent ADDQ with #n.L
ADDA ⁺	WL	s,An	-----	s e s s s s s s s s s s s	- - - - - - - - - -	$s + An \rightarrow An$	Add address (W sign-extended to L)
ADDI ⁺	BWL	#n,d	*****	d - d d d d d d d d d - - s	- - - - - - - - - -	$\#n + d \rightarrow d$	Add immediate to destination
ADDQ ⁺	BWL	#n,d	*****	d d d d d d d d d d d - - s	- - - - - - - - - -	$\#n + d \rightarrow d$	Add quick immediate (#n range: 1 to 8)
ADDX	BWL	Dy,Dx -(Ay)-:(Ax)	*****	e - - - - - d' - - - - -	- - - - - - - - - -	$Dy + Dx + X \rightarrow Dx$ $-(Ay) + -(Ax) + X \rightarrow -(Ax)$	Add source and eXtend bit to destination
AND ⁺	BWL	s,Dn Dn,d	---00	e - s s s s s s s s s s s ⁺ e - 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 ⁺	BWL	#n,d	---00	d - d d d d d d d d d - - s	- - - - - - - - - -	$\#n \text{ AND } d \rightarrow d$	Logical AND immediate to destination
ANDI ⁺	B	#n,CCR	=====	- - - - - - - - - -	- - - - - - - - - -	$\#n \text{ AND CCR} \rightarrow \text{CCR}$	Logical AND immediate to CCR
ANDI ⁺	W	#n,SR	=====	- - - - - - - - - -	- - - - - - - - - -	$\#n \text{ AND SR} \rightarrow \text{SR}$	Logical AND immediate to SR (Privileged)
ASL	BWL	Dx,Dy	*****	e - - - - - d - - - - -	- - - - - - - - - -		Arithmetic shift Dy by Dx bits left/right
ASR	W	#n,Dy d	*****	d - - - - - d - - - - -	- - - - - - - - - -		Arithmetic shift Dy #n bits L/R (#n: 1 to 8) Arithmetic shift d 1 bit left/right (W only)
Bcc	BW ²	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	---*	e' - d d d d d d d d d - - s d' - d d d d d d d d d - - s	- - - - - - - - - -	NOT(bit number of d) \rightarrow Z NOT(bit n of d) \rightarrow bit n of d	Set Z with state of specified bit in d then invert the bit in d
BCLR	B L	Dn,d #n,d	---*	e' - d d d d d d d d d - - s d' - d d d d d d d d d - - s	- - - - - - - - - -	NOT(bit number of d) \rightarrow Z 0 \rightarrow bit number of d	Set Z with state of specified bit in d then clear the bit in d
BRA	BW ²	address ²	-----	- - - - - - - - - -	- - - - - - - - - -	address \rightarrow PC	Branch always (8 or 16-bit \pm offset to addr)
BSET	B L	Dn,d #n,d	---*	e' - d d d d d d d d d - - s d' - d d d d d d d d d - - s	- - - - - - - - - -	NOT(bit n of d) \rightarrow Z 1 \rightarrow bit n of d	Set Z with state of specified bit in d then set the bit in d
BSR	BW ²	address ²	-----	- - - - - - - - - -	- - - - - - - - - -	PC \rightarrow -(SP); address \rightarrow PC	Branch to subroutine (8 or 16-bit \pm offset)
BTST	B L	Dn,d #n,d	---*	e' - d d d d d d d d d - - s d' - d d d d d d d d d - - s	- - - - - - - - - -	NOT(bit Dn of d) \rightarrow Z NOT(bit #n 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 ⁺	- - - - - - - - - -	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 - - -	- - - - - - - - - -	0 \rightarrow d	Clear destination to zero
CMP ⁺	BWL	s,Dn	-----	e s ⁺ s s s s s s s s s s s s ⁺	- - - - - - - - - -	set CCR with Dn - s	Compare Dn to source
CMPA ⁺	WL	s,An	-----	s e s s s s s s s s s s s s s	- - - - - - - - - -	set CCR with An - s	Compare An to source
CMPI ⁺	BWL	#n,d	-----	d - d d d d d d d d d - - s	- - - - - - - - - -	set CCR with d - #n	Compare destination to #n
CMPM ⁺	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 <= 0 then addr \rightarrow 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 ⁺	- - - - - - - - - -	$Dn \div \pm 16\text{bit } s \rightarrow \pm Dn$	Dn = [16-bit remainder, 16-bit quotient]
DIVU	W	s,Dn	----*0	e - 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-bit remainder, 16-bit quotient]
EOR ⁺	BWL	Dn,d	---00	e - d d d d d d d d d - - s ⁺	- - - - - - - - - -	$Dn \text{ XOR } d \rightarrow d$	Logical exclusive OR Dn to destination
EORI ⁺	BWL	#n,d	---00	d - d d d d d d d d d - - s	- - - - - - - - - -	$\#n \text{ XOR } d \rightarrow d$	Logical exclusive OR #n to destination
EORI ⁺	B	#n,CCR	=====	- - - - - - - - - -	- - - - - - - - - -	$\#n \text{ XOR CCR} \rightarrow \text{CCR}$	Logical exclusive OR #n to CCR
EORI ⁺	W	#n,SR	=====	- - - - - - - - - -	- - - - - - - - - -	$\#n \text{ XOR SR} \rightarrow \text{SR}$	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 - - - - - - - - - -	- - - - - - - - - -	$Dn.B \rightarrow Dn.W \mid Dn.W \rightarrow Dn.L$	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 - - - -	$\uparrow d \rightarrow \text{PC}$	Jump to effective address of destination
JSR		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 - - - - - e s - - - -	- e s - - - - - e 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 Logical shift Dy, #n bits L/R (#n: 1 to 8)
LSR	W	d	***0*	d - - - - - d - - - - -	- - - - - - - - - -		Logical shift d 1 bit left/right (W only)
MOVE ⁺	BWL	s,d	---00	e s ⁺ e e e e e e e e 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	- - - - - - - - - -	$s \rightarrow \text{CCR}$	Move source to Condition Code Register
MOVE	W	s,SR	=====	s - s s s s s s s s s s s s s	- - - - - - - - - -	$s \rightarrow \text{SR}$	Move source to Status Register (Privileged)
MOVE	W	SR,d	-----	d - d d d d d d d d d - - -	- - - - - - - - - -	$\text{SR} \rightarrow d$	Move Status Register to destination
MOVE	L	USP,An An,USP	-----	- d - - - - - - s - - - - -	- 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) (iAn) (iAn,Rn) abs.W abs.L (iPC) (iPC,Rn) #n			

APPENDIX 1: M68K Datasheet (continue)

Opcode	Size	Operand	CCR	Effective Address	s=source, d=destination, e=either, i=displacement	Operation	Description
	BWL	s,d	XNZVVC	Dn An (An) (An)+ -(An) (iAn) (iAn,Rn) abs.W abs.L (iPC) (iPC,Rn) #n			
MOVEA ⁺	WL	s,An	-----	s e s s s s s s s s s s s s s s	s → An	Move source to An (MOVE s,An use MOVEA)	
MOVEM ⁺	WL	Rn-Rn,d s,Rn-Rn	-----	- - d - d d 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	WL	Dn,(iAn) (iAn),Dn	-----	s - - - - d - - - - - - - - - -	Dn → (iAn)...(i+2,An)...(i+4,An) (iAn) → Dn...(i+2,An)...(i+4,An)	Move Dn to/from alternate memory bytes (Access only even or odd addresses)	
MOVEQ ⁺	L	#n,Dn	---*00	d - - - - - - - - - - - - - - - -	#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 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 s s s s	16bit s * 16bit Dn → Dn	Multiply unsg'd 16-bit; result: unsg'd 32-bit	
NBCD	B	d	*U*U*	d - d d d d d d d d d d - - - -	0 - d ₀ - X → d	Negate BCD with eXtend, BCD result	
NEG	BWL	d	*****	d - 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 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 d d d - - - -	NOT(d) → d	Logical NOT destination (1's complement)	
OR ⁺	BWL	s,Dn Dn,Dn	---*00	e - s s s s 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 d d d - - - -	#n OR d → d	Logical OR #n to destination	
ORI ⁺	B	#n,CCR	=====	- - - - - - - - - - - - - - - -	#n OR CCR → CCR	Logical OR #n to CCR	
ORI ⁺	W	#n,SR	=====	- - - - - - - - - - - - - - - -	#n OR SR → SR	Logical OR #n to SR (Privileged)	
PEA	L	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)	
ROL	BWL	Dx,Dy	---*0*	e - - - - - - - - - - - - - - - -		Rotate Dy, Dx bits left/right (without X)	
ROR	W	#n,Dy	-----	d - - d d d d d d d d d d - - - -		Rotate Dy, #n bits left/right (#n: 1 to 8) Rotate d 1-bit left/right (W only)	
ROXL	BWL	Dx,Dy	---*0*	e - - - - - - - - - - - - - - - -		Rotate Dy, Dx bits L/R, X used then updated	
ROXR	W	#n,Dy	-----	d - - d d d d d d d d d d - - - -		Rotate Dy, #n bits left/right (#n: 1 to 8) 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 -(Ay),-(Ax)	*U*U*	e - - - - - - - - - - - - - - - -	Dx ₀ - Dy ₀ - X → Dx ₀ -(Ax) ₀ - -(Ay) ₀ - X → -(Ax) ₀	Subtract BCD source and eXtend bit from destination, BCD result	
Sec	B	d	-----	d - d d d d d d d d d d - - - -	If cc is true then 1's → d else 0's → d	If cc true then d.B = 11111111 else d.B = 00000000	
STOP		#n	=====	- - - - - - - - - - - - - - - -	#n → SR; STOP	Move #n to SR, stop processor (Privileged)	
SUB ⁺	BWL	s,Dn Dn,Dn	*****	e s s s 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 ⁺	WL	s,An	-----	s e s s s s 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 d d - - - -	d - #n → d	Subtract immediate from destination	
SUBQ ⁺	BWL	#n,d	*****	d d d d d d d d d d d d - - - -	d - #n → d	Subtract quick immediate (#n range: 1 to 8)	
SUBX	BWL	Dy,Dx -(Ay),-(Ax)	*****	e - - - - - 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 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); (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 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	XNZVVC	Dn An (An) (An)+ -(An) (iAn) (iAn,Rn) abs.W abs.L (iPC) (iPC,Rn) #n			

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

An Address register (16/32-bit, n=0-7)
Dn Data register (8/16/32-bit, n=0-7)
Rn any data or address register
s Source, **d** Destination
e Either source or destination
#n Immediate data, **i** Displacement
BCD Binary Coded Decimal
↑ Effective address
1 Long only; all others are byte only
2 Assembler calculates offset
3 Branch sizes: **B** or **S** -128 to +127 bytes, **.W** or **.L** -32768 to +32767 bytes
4 Assembler automatically uses A, I, Q or M form if possible. Use #n.L to prevent Quick optimization
SSP Supervisor Stack Pointer (32-bit)
USP User Stack Pointer (32-bit)
SP Active Stack Pointer (same as A7)
PC Program Counter (24-bit)
SR Status Register (16-bit)
CCR Condition Code Register (lower 8-bits of SR)
N negative, **Z** zero, **V** overflow, **C** carry, **X** extend
 * set according to operation's result, = set directly
 - not affected, **O** cleared, **I** set, **U** undefined

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

Distributed under the GNU general public use license.