

Midterm Exam S3

Computer Architecture

Duration: 1 hr 30 min

Write answers only on the answer sheet.

Exercise 1 (5 points)

Complete the table shown on the answer sheet. Write down the new values of the registers (except the PC) and memory that are modified by the instructions. Use the hexadecimal representation. Memory and registers are reset to their initial values for each instruction.

Initial values: D0 = \$FFFF0010 A0 = \$00005000 PC = \$00006000
 D1 = \$10000002 A1 = \$00005008
 D2 = \$FFFFFFF0 A2 = \$00005010

\$005000 54 AF 18 B9 E7 21 48 C0
 \$005008 C9 10 11 C8 D4 36 1F 88
 \$005010 13 79 01 80 42 1A 2D 49

Exercise 2 (4 points)

Complete the table shown on the answer sheet. Give the result of the additions and the values of the N, Z, V and C flags.

Exercise 3 (2 points)

Let us consider the following programs. Complete the table shown on the answer sheet.

```
move.l  #$76543210,d1
ror.l   #8,d1
ror.b   #4,d1
swap    d1
ror.b   #4,d1
```

```
move.l  #$76543210,d2
ror.b    #4,d2
ror.w    #8,d2
ror.b    #4,d2
ror.w    #8,d2
```

Exercise 4 (3 points)

Answer the questions on the answer sheet.

Exercise 5 (6 points)

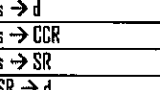
Let us consider the following program. Complete the table shown on the answer sheet.

Main	move.l #158f,d7
next1	moveq.l #1,d1 tst.b d7 bpl next2 moveq.l #2,d1
next2	moveq.l #1,d2 tst.l d7 bmi next3 moveq.l #2,d2
next3	clr.l d3 move.l #87654321,d0
loop3	addq.l #1,d3 subq.w #1,d0 bne loop3
next4	clr.l d4 move.w #5aa,d0
loop4	addq.l #1,d4 dbra d0,loop4 ; DBRA = DBF
next5	moveq.l #1,d5 cmp.b #542,d7 bgt next6 moveq.l #2,d5
next6	moveq.l #1,d6 cmp.b #542,d7 bls quit moveq.l #2,d6
quit	illegal

EASy68K Quick Reference v1.8

<http://www.wowgweb.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) (An) (An,Rn) abs.W abs.L (i.PC) (i.PC,Rn) #n				
ABCD	B	Dy,Dx -(Ay).-(Ax)	*0*0*	e - - - - - - - e	- - - - - - - -	$Dy + Dx + X \rightarrow Dx$ $-(Ay)_D + -(Ax)_D + X \rightarrow -(Ax)_D$	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 e d ¹ d d d d	s s s s s s s s 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	$s + An \rightarrow An$	Add address (W sign-extended to .L)	
ADDI ⁴	BWL	#n,d	*****	d - d d d d d d	d d d d 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 d d d d	$\#n + d \rightarrow d$	Add quick immediate (#n range: 1 to 8)	
ADDX	BWL	Dy,Dx -(Ay).-(Ax)	*****	e - - - - e - - - 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 e - d d d d d	s s s s s s s s 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 d d d	$\#n \text{ AND } d \rightarrow d$	Logical AND immediate to destination	
ANDI ⁴	B	#n.CCR	0000 0000	- - - - -	- - - - -	$\#n \text{ AND } CCR \rightarrow CCR$	Logical AND immediate to CCR	
ANDI ⁴	W	#n.SR	0000 0000	- - - - -	- - - - -	$\#n \text{ AND } SR \rightarrow SR$	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 d d d - - d d d d	- - - - - - - -	- - - - - - - -		Arithmetic shift Dy #n bits L/R (#n: 1 to 8) Arithmetic shift ds 1 bit left/right (W only)
Bcc	BW ⁴	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 ¹ - d d d d d d ¹ - 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	$\text{NOT}(\text{bit number of } d) \rightarrow Z$ $\text{NOT}(\text{bit } n \text{ of } d) \rightarrow \text{bit } n \text{ of } d$	Set Z with state of specified bit in d then invert the bit in d
BCLR	B L	Dn,d #n,d	---*----	d ¹ - d d d d d	d d d 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 ⁴	address ²	-----	- - - - -	- - - - -	- - - - -	address \rightarrow PC	Branch always (8 or 16-bit \pm offset to addr)
BSET	B L	Dn,d #n,d	---*----	e ¹ - d d d d d d ¹ - d d d d	d d d 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
BSR	BW ⁴	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 ¹ - d d d d d d ¹ - d d d d	d d d 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	---0000	e - s s s s 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 D and upper bound (s)
CLR	BWL	d	-0100	d - d d d d d	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 s s s	set CCR with $Dn - s$	Compare Dn to source
CMPI ⁴	WL	s.An	*****	s e 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 An to source
CMPI ⁴	BWL	#n,d	*****	d - d d d d d d	d d d d d d	d d d d d	set CCR with $d - \#n$	Compare destination to #n
CMPI ⁴	BWL	(Ay).-(Ax)+	*****	- - - e - - - -	- - - - -	- - - - -	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 addr \rightarrow PC }	Test condition, decrement and branch (16-bit \pm offset to address)
DIVS	W	s,Dn	---*00	e - s s s s 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	---*00	e - s s s s 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 d d	d d d d d	$Dn \text{ XOR } d \rightarrow d$	Logical exclusive OR Dn to destination
EORI ⁴	BWL	#n,d	---*00	d - d d d d d d	d d d d d d	d d d d d	$\#n \text{ XOR } d \rightarrow d$	Logical exclusive OR #n to destination
EORI ⁴	B	#n.CCR	0000 0000	- - - - -	- - - - -	- - - - -	$\#n \text{ XOR } CCR \rightarrow CCR$	Logical exclusive OR #n to CCR
EORI ⁴	W	#n.SR	0000 0000	- - - - -	- - - - -	- - - - -	$\#n \text{ XOR } SR \rightarrow SR$	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 - - - - -	- - - - -	- - - - -	$Dn.B \rightarrow Dn.W \mid Dn.W \rightarrow Dn.L$	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 d d d	d d d d d	$Td \rightarrow PC$	Jump to effective address of destination
JSR		d		- - d - - - -	d d d d d d	d d d d d	PC \rightarrow -(SP); $Td \rightarrow PC$	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	$Ts \rightarrow An$	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	***0*	e - - - - d - - - -	- - - - - - - -	- - - - - - - -		Logical shift Dy, Dx bits left/right
LSR	W	#n,Dy d		- - d d d d d - - d d d d	- - - - - - - -	- - - - - - - -		Logical shift Dy, #n bits L/R (#n: 1 to 8) 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 e e e s s	s s s s s s	$s \rightarrow d$	Move data from source to destination
MOVE	W	s.CCR	0000 0000	s - s s s 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	0000 0000	s - s s s 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 d d	d d d d d	$SR \rightarrow d$	Move Status Register to destination
MOVE	L	USP,An		- d - - - - -	- - - - -	- - - - -	$USP \rightarrow An$	Move User Stack Pointer to An (Privileged)
	BWL	An,USP		- s - - - - -	- - - - -	- - - - -	$An \rightarrow USP$	Move An to User Stack Pointer (Privileged)

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		
MOVEA ¹	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 ¹	WL	Rn-Rn,d	-----	-	-	d	-	d	d	d	d	-	-	-	-	Registers → d	Move specified registers to/from memory (W source is sign-extended to L for Rn)
MOVEP	WL	Dn,(iAn)	-----	s	-	-	-	-	s	-	-	-	-	-	-	Dn → (iAn)...(i+2An)...(i+4An)	Move Dn to/from alternate memory bytes (Access only even or odd addresses)
MOVED ¹	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 ±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 16bit s * 16bit Dn → Dn	Multiply unisg'd 16-bit; result: unisg'd 32-bit
NBCD	B	d	*U*U*	d	-	d	d	d	d	d	d	-	-	-	-	0 - dg - X → d	Negate BCD with eXtend, BCD result
NEG	BWL	d	*****	d	-	d	d	d	d	d	d	d	-	-	-	0 - d → d	Negate destination (2's complement)
NEGX	BWL	d	*****	d	-	d	d	d	d	d	d	d	-	-	-	0 - d - X → d	Negate destination with eXtend
NOP			---	-	-	-	-	-	-	-	-	-	-	-	-	None	No operation occurs
NOT	BWL	d	---*00	d	-	d	d	d	d	d	d	d	-	-	-	NOT(d) → d	Logical NOT destination (1's complement)
OR ¹	BWL	s,Dn	---*00	e	-	s	s	s	s	s	s	s	s	s	s	s OR Dn → Dn	Logical OR
		Dn,d		e	-	d	d	d	d	d	d	d	-	-	-	Dn OR d → d	(ORI is used when source is #n)
ORI ¹	BWL	#n,d	---*00	d	-	d	d	d	d	d	d	d	-	-	s	#n OR d → d	Logical OR #n to destination
ORI ¹	B	#n,CCR	---	-	-	-	-	-	-	-	-	-	-	-	s	#n OR CCR → CCR	Logical OR #n to CCR
ORI ¹	W	#n,SR	---	-	-	-	-	-	-	-	-	-	-	-	s	#n OR SR → SR	Logical OR #n to SR (Privileged)
PEA	L	s	---	-	-	s	-	-	s	s	s	s	s	s	-	↑s → -(SP)	Push effective address of s onto stack
RESET			---	-	-	-	-	-	-	-	-	-	-	-	-	Assert RESET Line	Issue a hardware RESET (Privileged)
ROL	BWL	Dx,Dy	---*0*	e	-	-	-	-	-	-	-	-	-	-	-		
ROR	BWL	#n,Dy	---	d	-	-	-	-	-	-	-	-	-	-	s		
	W	d		-	-	d	d	d	d	d	d	d	-	-	-		
ROXL	BWL	Dx,Dy	---*0*	e	-	-	-	-	-	-	-	-	-	-	-		
ROXR	BWL	#n,Dy	---	d	-	-	-	-	-	-	-	-	-	-	s		
	W	d		-	-	d	d	d	d	d	d	d	-	-	-		
RTE			---	-	-	-	-	-	-	-	-	-	-	-	-	(SP)+ → SR; (SP)+ → PC	Return from exception (Privileged)
RTR			---	-	-	-	-	-	-	-	-	-	-	-	-	(SP)+ → CCR; (SP)+ → PC	Return from subroutine and restore CCR
RTS			---	-	-	-	-	-	-	-	-	-	-	-	-	(SP)+ → PC	Return from subroutine
SBCD	B	Dy,Dx	*U*U*	e	-	-	-	-	-	-	-	-	-	-	-	Dx ₀ - Dy ₀ - X → Dx ₀	Subtract BCD source and eXtend bit from destination, BCD result
		-(Ay),-(Ax)		-	-	-	-	e	-	-	-	-	-	-	-	-(Ax) ₀ - (Ay) ₀ - X → -(Ax) ₀	
Scc	B	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	---	-	-	-	-	-	-	-	-	-	-	-	s	#n → SR; STOP	Move #n to SR, stop processor (Privileged)
SUB ¹	BWL	s,Dn	*****	e	s	s	s	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)
		Dn,d		e	d	d	d	d	d	d	d	d	-	-	-	d - Dn → d	
SUBA ¹	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 ¹	BWL	#n,d	*****	d	-	d	d	d	d	d	d	d	-	-	s	d - #n → d	Subtract immediate from destination
SUBQ ¹	BWL	#n,d	*****	d	d	d	d	d	d	d	d	d	-	-	s	d - #n → d	Subtract quick immediate (#n range: 1 to 8)
SUBX	BWL	Dy,Dx	*****	e	-	-	-	-	-	-	-	-	-	-	-	Dx - Dy - X → Dx	Subtract source and eXtend bit from destination
		-(Ay),-(Ax)		-	-	-	-	e	-	-	-	-	-	-	-	-(Ax) - (Ay) - X → -(Ax)	
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
	BWL	s,d	XNZVC	Dn	An	(An)	(An)+	-(An)	(iAn)	(iAn,Rn)	abs.W	abs.L	(iPC)	(iPC,Rn)	#n		

Condition Tests (+ OR, 1 NOT, ● XOR; * Unsigned, * Alternate cc)				
cc	Condition	Test	cc	Condition
T	true	I	VC	overflow clear
F	false	0	VS	overflow set
HI*	higher than	(K + Z)	PL	plus
LS*	lower or same	C + Z	MI	minus
HS*, CC*	higher or same	IC	GE	greater or equal
LO*, CS*	lower than	C	LT	less than
NE	not equal	IZ	GT	greater than
EQ	equal	Z	LE	less or equal

An Address register (16/32-bit, n=0-7)
Dn Data register (8/16/32-bit, n=0-7)
Rn any data or address register
s Source, **d** Destination
e Either source or destination
#n Immediate data, **i** Displacement
BCD Binary Coded Decimal
↑ Effective address
1 Long only; all others are byte only
2 Assembler calculates offset
3 Branch sizes: **B** or **S** -128 to +127 bytes, **W** or **L** -32768 to +32767 bytes
4 Assembler automatically uses **A**, **I**, **Q** or **M** form if possible. Use **#n.L** to prevent Quick optimization

SSP Supervisor Stack Pointer (32-bit)
USP User Stack Pointer (32-bit)
SP Active Stack Pointer (same as A7)
PC Program Counter (24-bit)

SR Status Register (16-bit)
CCR Condition Code Register (lower 8-bits of SR)
N negative, **Z** zero, **V** overflow, **C** carry, **X** extend
 * set according to operation's result, = set directly
 - not affected, 0 cleared, 1 set, U undefined

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

Distributed under the GNU general public use license.

Last name: First name: Group:

ANSWER SHEET TO BE HANDED IN**Exercise 1**

Instruction	Memory	Register
Example	\$005000 54 AF 00 40 E7 21 48 C0	A0 = \$00005004 A1 = \$0000500C
Example	\$005008 C9 10 11 C8 D4 36 FF 88	No change
MOVE.W #\$500A, -(A1)		
MOVE.W \$500A, -2(A1)		
MOVE.L \$500A, -(A1)		
MOVE.B 5(A1), 3(A2, D2.L)		
MOVE.L -4(A1), -16(A2, D0.W)		

Exercise 2

Operation	Size (bits)	Result (hexadecimal)	N	Z	V	C
\$5A + \$35	8					
\$5A + \$35	16					
\$7F8C + \$FFFF	16					
\$FFFFFFFF0 + \$00000010	32					

Exercise 3

Values of registers after the execution of the program. Use the 32-bit hexadecimal representation.	
D1 = \$	D2 = \$

Exercise 4

Question	Answer
How many data registers does the 68000 have?	
How many address registers does the 68000 have?	
How many program counters does the 68000 have?	
How many stack pointers does the 68000 have?	
How many status registers does the 68000 have?	
How many levels of privilege does the 68000 have?	

Exercise 5

Values of registers after the execution of the program. Use the 32-bit hexadecimal representation.		
D1 = \$	D3 = \$	D5 = \$
D2 = \$	D4 = \$	D6 = \$