

Midterm Exam S3

Computer Architecture

Duration: 1 hr. 30 min.

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 = \$0004FFFF A0 = \$00005000 PC = \$00006000
 D1 = \$FFFF0005 A1 = \$00005008
 D2 = \$FFFFFFFE 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 (3 points)

Write a few instructions that modify **D1** so that it takes the values given on the answer sheet. For each case, the initial value of **D1** is \$76543210. **Use ROR, ROL or SWAP only.** Answer on the answer sheet.

Exercise 4 (2 points)

Answer the questions on the answer sheet.

Exercise 5 (6 points)

Let us consider the following program:

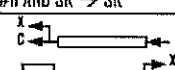
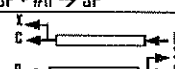
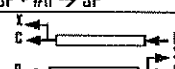
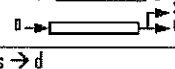
Main	move.l	#\$23456789,d7	
next1	moveq.l	#1,d1	
	tst.b	d7	
	bmi	next2	
	moveq.l	#2,d1	
next2	moveq.l	#1,d2	
	tst.w	d7	
	bpl	next3	
	moveq.l	#2,d2	
next3	clr.l	d3	
	move.w	#\$4321,d0	
loop3	addq.l	#1,d3	
	subq.b	#1,d0	
	bne	loop3	
next4	clr.l	d4	
	move.w	#\$44,d0	
loop4	addq.l	#1,d4	
	dbra	d0,loop4	; DBRA = DBF
next5	clr.l	d5	
	moveq.l	#10,d0	
loop5	addq.l	#1,d5	
	addq.l	#1,d0	
	cmpi.l	#30,d0	
	bne	loop5	
next6	moveq.l	#1,d6	
	cmp.b	#\$70,d7	
	blt	quit	
	moveq.l	#2,d6	
quit	illegal		

Complete the table shown on the [answer sheet](#).

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)	(iAn)	(iAn,Rn)	abs.W	abs.L	(i.PC)	(i.PC,Rn)	#n		
ABCD	B	Dy,Dx -(Ay),-(Ax)	*U*U*	e	-	-	-	-	-	-	-	-	-	-	-	$Dy + Dx_{10} + X \rightarrow Dx_{10}$ $-(Ay)_{10} + -(Ax)_{10} + X \rightarrow -(Ax)_{10}$	Add BCD source and eXtend bit to destination, BCD result	
ADD ⁺	BWL	s,Dn Dn,d	*****	e	s	s	s	s	s	s	s	s	s	s	s ⁺	$s + Dn \rightarrow Dn$ $Dn + d \rightarrow d$	Add binary (ADDI or ADDQ is used when source is #n. Prevent ADDQ with #n.L)	
ADDA ⁺	WL	s,An	-----	s	e	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	-	-	s	$\#n + d \rightarrow d$	Add immediate to destination	
ADDQ ⁺	BWL	#n,d	*****	d	d	d	d	d	d	d	d	d	-	-	s	$\#n + d \rightarrow d$	Add quick immediate (#n range: l 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 ⁺	BWL	s,Dn Dn,d	---00	e	-	s	s	s	s	s	s	s	s	s	s ⁺	$s \text{ AND } Dn \rightarrow Dn$ $Dn \text{ AND } d \rightarrow d$	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	-	-	s	$\#n \text{ AND } d \rightarrow d$	Logical AND immediate to destination	
ANDI ⁺	B	#n,CCR	00 00 00 00	-	-	-	-	-	-	-	-	-	-	-	s	$\#n \text{ AND } CCR \rightarrow CCR$	Logical AND immediate to CCR	
ANDI ⁺	W	#n,SR	00 00 00 00	-	-	-	-	-	-	-	-	-	-	-	s	$\#n \text{ AND } SR \rightarrow SR$	Logical AND immediate to SR (Privileged)	
ASL	BWL	Dx,Dy	*****	e	-	-	-	-	-	-	-	-	-	-	-		Arithmetic shift Dy by Dx bits left/right	
ASR		#n,Dy d		d	-	d	d	d	d	d	d	d	d	-	-	s		Arithmetic shift Dy #n bits L/R (#n: l to 8) Arithmetic shift d l bit left/right (.W only)
Bcc	BW ^d	address ²	-----	-	-	-	-	-	-	-	-	-	-	-	-	if cc true then address \rightarrow PC	Branch conditionally (cc table on back) (B or l6-bit \pm offset to address)	
BCNG	B L	Dn,d #n,d	---*-	e ¹	-	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	---*-	e ¹	-	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 (B or l6-bit \pm offset to addr)	
BSET	B L	Dn,d #n,d	---*-	e ¹	-	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 ^d	address ²	-----	-	-	-	-	-	-	-	-	-	-	-	-	PC \rightarrow -(SP); address \rightarrow PC	Branch to subroutine (B or l6-bit \pm offset)	
BTST	B L	Dn,d #n,d	---*-	e ¹	-	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	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	-	-	-	$0 \rightarrow d$	Clear destination to zero	
CMP ⁺	BWL	s,Dn	*****	e	s ⁺	s	s	s	s	s	s	s	s	s	s	set CCR with $Dn - s$	Compare Dn to source	
CMPI ⁺	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 ⁺	BWL	#n,d	*****	d	-	d	d	d	d	d	d	d	-	-	s	set CCR with $d - \#n$	Compare destination to #n	
CMPI ⁺	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 (l6-bit \pm offset to address)	
DIVS	W	s,Dn	---*0	e	-	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	$32\text{bit } Dn / 16\text{bit } s \rightarrow Dn$	$Dn = [16\text{-bit remainder}, 16\text{-bit quotient}]$	
EDR ⁺	BWL	Dn,d	---00	e	-	d	d	d	d	d	d	d	-	-	s ⁺	$Dn \text{ XOR } d \rightarrow d$	Logical exclusive OR Dn to destination	
EDRI ⁺	BWL	#n,d	---00	d	-	d	d	d	d	d	d	d	-	-	s	$\#n \text{ XOR } d \rightarrow d$	Logical exclusive OR #n to destination	
EDRI ⁺	B	#n,CCR	00 00 00 00	-	-	-	-	-	-	-	-	-	-	-	s	$\#n \text{ XOR } CCR \rightarrow CCR$	Logical exclusive OR #n to CCR	
EDRI ⁺	W	#n,SR	00 00 00 00	-	-	-	-	-	-	-	-	-	-	-	s	$\#n \text{ XOR } SR \rightarrow SR$	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	-	$\uparrow d \rightarrow PC$	Jump to effective address of destination	
JSR		d	-----	-	-	d	-	-	d	d	d	d	d	d	-	PC \rightarrow -(SP); $\uparrow d \rightarrow PC$	push PC, jump to subroutine at address d	
LEA	L	s,An	-----	-	e	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	***0*	e	-	-	-	-	-	-	-	-	-	-	-		Logical shift Dy, Dx bits left/right	
LSR		#n,Dy d		d	-	d	d	d	d	d	d	d	d	-	-	s		Logical shift Dy, #n bits L/R (#n: l to 8) Logical shift d l bit left/right (.W only)
MOVE ⁺	BWL	s,d	---00	e	s ⁺	e	e	e	e	e	e	e	s	s	s ⁺	$s \rightarrow d$	Move data from source to destination	
MOVE	W	s,CCR	00 00 00 00	s	-	s	s	s	s	s	s	s	s	s	s	$s \rightarrow CCR$	Move source to Condition Code Register	
MOVE	W	s,SR	00 00 00 00	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	-	-	-	$SR \rightarrow d$	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	(i.PC)	(i.PC,Rn)	#n			
MOVEA ⁴	WL	s,An	-----	s	e	s	s	s	s	s	s	s	s	s	s	s	s → An	Move source to An (MOVE sAn use MOVEA)
MOVEM ⁴	WL	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	WL	Dn,(iAn) (iAn),Dn	-----	s	-	-	-	-	d	-	-	-	-	-	-	-	Dn → (iAn)...(i+2An)...(i+4A. (iAn) → Dn...(i+2An)...(i+4A.	Move Dn to/from alternate memory bytes (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	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 unsig'd 16-bit; result: unsig'd 32-bit
NBCD	B	d	*0*U*	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	-	-	-	-	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
NDP			-----	-	-	-	-	-	-	-	-	-	-	-	-	-	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 Dn,d	-***00	e	-	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 ⁴	BWL	#n,d	-***00	d	-	d	d	d	d	d	d	d	-	-	-	s	#n OR d → d	Logical OR #n to destination
ORI ⁴	B	#n,CCR	000000	-	-	-	-	-	-	-	-	-	-	-	-	s	#n OR CCR → CCR	Logical OR #n to CCR
ORI ⁴	W	#n,SR	000000	-	-	-	-	-	-	-	-	-	-	-	-	s	#n OR SR → SR	Logical OR #n to SR (Privileged)
PEA	L	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	-	-	-	-	-	-	-	-	-	-	-	-		Rotate Dy, Dx bits left/right (without X)
RDR	W	#n,Dy d	-***0*	d	-	-	-	-	-	-	-	-	-	-	-	s		Rotate Dy, #n bits left/right (#n: 1 to 8) Rotate d 1-bit left/right (W only)
RDXL	BWL	Dx,Dy	-***0*	e	-	-	-	-	-	-	-	-	-	-	-	-		Rotate Dy, Dx bits L/R, X used then updated
RDXR	W	#n,Dy d	-***0*	d	-	-	-	-	-	-	-	-	-	-	-	s		Rotate Dy, #n bits left/right (#n: 1 to 8) Rotate destination 1-bit left/right (W only)
RTE			000000	-	-	-	-	-	-	-	-	-	-	-	-	-	(SP)+ → SR; (SP)+ → PC	Return from exception (Privileged)
RTR			000000	-	-	-	-	-	-	-	-	-	-	-	-	-	(SP)+ → CCR; (SP)+ → PC	Return from subroutine and restore CCR
RTS			-----	-	-	-	-	-	-	-	-	-	-	-	-	-	(SP)+ → PC	Return from subroutine
SBCD	B	Dy,Dx -(Ay),-(Ax)	*0*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
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	00000000	-	-	-	-	-	-	-	-	-	-	-	-	s	#n → SR; STOP	Move #n to SR, stop processor (Privileged)
SUB ⁴	BWL	s,Dn Dn,d	*****	e	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 ⁴	WL	s,An	-----	s	e	s	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 -(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
	BWL	s,d	XNZVC	Dn	An	(An)	(An)+	-(An)	(iAn)	(iAn,Rn)	abs.W	abs.L	(i.PC)	(i.PC,Rn)	#n			

Condition Tests (→ OR, !NOT, ⊕ XOR, * Unsigned, * Alternate cc)					
cc	Condition	Test	cc	Condition	Test
I	true	I	VC	overflow clear	IV
F	false	D	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	I(N ⊕ V)
LO*, CS*	lower than	C	LT	less than	(N ⊕ V)
NE	not equal	IZ	GT	greater than	I[(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, 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.L (A2)+,(A0)+		
MOVE.L 4(A2),4(A0)		
MOVE.B \$500A,-1(A1,D0.W)		
MOVE.L #\$500A,-5(A1,D1.W)		
MOVE.W \$500A,-(A1)		

Exercise 2

Operation	Size (bits)	Result (hexadecimal)	N	Z	V	C
\$F0 + \$11	8					
\$F0 + \$11	16					
\$8000 + \$8000	16					
\$40000000 + \$80000000	32					

Exercise 3

Final value of **D1** : **\$76542301**. Use four lines of instructions at the most.

Final value of **D1** : **\$54231067**. Use four lines of instructions at the most.

Exercise 4

Question	Answer
Give two assembler directives.	
How many status register does the 68000 have?	
What is the size of the CCR register?	
Which 68000 mode has limited privileges?	

Exercise 5

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