Final Exam S3 Computer Architecture

Duration: 1 hr 30 min

Write answers only on the answer sheet.

Exercise 1 (4 points)

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

Initial values: D0 = \$FFFF0020 A0 = \$00005000 PC = \$00006000 D1 = \$12340004 A1 = \$00005008 D2 = \$FFFFFF7 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 (3 points)

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

Exercise 3 (4 points)

Let us consider the following program. Complete the table shown on the <u>answer sheet</u>.

Main	move.l	#\$5A9500,d7	
next1	moveq.l tst.b beq moveq.l	d7 next2	
next2	moveq.l cmpi.w bgt moveq.l	#\$95,d7 next3	
next3		#\$512,d0	
loop3	addq.l subq.b bne		
next4	clr.l move.l		
loop4	addq.l dbra	#1,d4	; DBRA = DBF
quit	illegal		

Exercise 4 (9 points)

All the questions in this exercise are independent. **Except for the output registers, none of the data or address registers must be modified when the subroutine returns.** For the whole exercise, a pair of ASCII characters is made up of two characters representing an integer between 0 and 99 (in base 10). For instance, the characters "08" ('0' and '8') represent the value 8; and the characters "42" ('4' and '2') represent the value 42. As a reminder, the ASCII code of the '0' character is equal to \$30.

Be careful. All the subroutines must contain 15 lines of instructions at the most.

1. Write the **ToAscii** subroutine that converts an integer into a pair of ASCII characters.

Input: **D0.L** holds an integer between 0 and 99 (in base 10).

A0.L points to a 2-byte buffer where the pair of ASCII characters must be written.

Example: If **D0** = \$0000002A (42 in base 10) and **A0.L** = \$00005000, then the address \$5000 will hold the value \$34 (ASCII code of '4') and the address \$5001 will hold the value \$32 (ASCII code of '2').

2. Write the **ToInt** subroutine that converts a pair of ASCII characters into an integer.

Input: **A0.L** points to a pair of ASCII characters to convert.

<u>Output</u>: **D0.B** holds the integer that is represented by the pair of ASCII characters.

Be careful, only **D0.B** must be modified (bits from 8 to 31 must not be modified).

Example:

Main	movea.l #ascii,a0 jsr ToInt illegal	; D0.B = \$2A (42 in base 10)	
ascii	dc.b "42"		

3. By using the **ToInt** subroutine, write the **GetSum** subroutine that returns the sum of two pairs of ASCII characters.

Input: **A0.L** points to a first pair of ASCII characters.

A1.L points to a second pair of ASCII characters.

<u>Output</u>: **D0.B** holds an integer that is the sum of the two pairs of ASCII characters. Be careful, only **D0.B** must be modified (bits from 8 to 31 must not be modified).

Example:

Main	movea.l #ascii1,a0 movea.l #ascii2,a1 jsr GetSum illegal	; D0.B = \$32 (50 in base 10)
ascii1 ascii2	dc.b "42" dc.b "08"	

Computer Architecture – EPITA – S3 – 2020/2021

		K Quic	k Ref	er	en	ice	v1.	8	htt	p://www	w.wo	wgw	ep.co	m/EAS	y68	K.htm Copyrigh	t © 2004-2007 By: Chuck Kelly
Opcode	Size	Operand	CCR	I	Effec	ctive	Addres	SS S=S	ource,	d=destina	ition, e	=eithe	r, i=dis	placemen	ıt	Operation	Description
	BWL	s,d	XNZVC	Dn	Ап	(An)	(An)+	-(An)	(i,An)	(i,An,Rn)	abs.W	abs.L	(i,PC)	(i,PC,Rn)	#n		
ABCD	8	Dy,Dx -(Ay),-(Ax)	*U*U*	B -	-	-	15. 12	- 8	-	121	-	-	2	10	ан) С	$\begin{array}{l} \mathbb{D} y_{10} + \mathbb{D} x_{10} + X \rightarrow \mathbb{D} x_{10} \\ -(\mathbb{A} y)_{10} + -(\mathbb{A} x)_{10} + X \rightarrow -(\mathbb{A} x)_{10} \end{array}$	Add BCD source and eXtend bit to destination, BCD result
ADD ⁴		s,Dn Dn,d	****	e e	s d ⁴	s d	s d	s d	s d	s d	s d	s d	8 -	S -	s ⁴	$s + Dn \rightarrow Dn$ Dn + d $\rightarrow d$	Add binary (ADDI or ADDQ is used when source is #n. Prevent ADDQ with #n.L)
ADDA ⁴		s,An		S	e	S	S	s s	S	s	S	S	5	8	2000		
ADDI ⁴		#n,d	****	d	-	d	d	d	d	d	d	d	-	<u>8</u>		$\#n + d \rightarrow d$	Add immediate to destination
		#n,u #n,d	****	d	d	d	d	d	d	d	d	d	-	-		#n+d → d	Add quick immediate (#n range: 1 to 8)
ADDX			****		-								-	-	8		Add source and eXtend bit to destination
		Dy,Dx -(Ay),-(Ax)		е -	-	-		- 8	-	-	2	-	2		-	$Dy + Dx + X \rightarrow Dx$ -(Ay) + -(Ax) + X \rightarrow -(Ax)	
ND ⁴	BWL	s,Dn Dn,d	-**00	e e	-	s d	s d	s d	s d	s d	s d	s d	5	8 -	s ⁴	s AND Dn \rightarrow Dn Dn 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	-	-	5	#n AND d \rightarrow d	Logical AND immediate to destination
ANDI 4		#n,CCR		-		-	-	-	-	-	-	-	-	-		#n AND CCR \rightarrow CCR	Logical AND immediate to CCR
ANDI ⁴		#n.SR		-	-	-	-	-	-	-	-	-	-	-	8	#n AND SR \rightarrow SR	Logical AND immediate to SR (Privileged)
ISL		Dx,Dy	****	-		-	-	-	-	-	-	-	-	-		Y -	
ASR		#n,Dy		e d	-	-	621	- - d		-	-	-	2	-	- S		Arithmetic shift Dy by Dx bits left/right Arithmetic shift Dy #n bits L/R (#n: 1 to 8)
10000	100000-0	d 7	-	-	-	d	d		d	d	d	d			्रः		Arithmetic shift ds 1 bit left/right (.W only)
Bcc	BM ₃	address ²		-		7	-	-	-		-	-	5	5 7 .		if cc true then address \rightarrow PC	Branch conditionally (cc table on back) (8 or 16-bit ± offset to address)
BCHG	BL	Dn,d	*	el	-	d	d	d	d	d	d	d	-	-	-	NDT(bit number of d) \rightarrow Z	Set Z with state of specified bit in d then
		#n,d		ď	-	d	d	d	d	d	d	d	-	-	8	NDT(bit n of d)→ bit n of d	invert the bit in d
BCLR	BL	Dn,d	*	e	-	d	d	d	d	d	d	d	-	-	14	NDT(bit number of d) $ ightarrow$ Z	Set Z with state of specified bit in d then
104		#n.d		ď	-	d	d	d	d	d	d	d	-			D → bit number of d	clear the bit in d
BRA		address ²		-	1.00	-	-	-	-	-	-	•	-	-	्रहर	address \rightarrow PC	Branch always (8 or 16-bit ± offset to addr
BSET		Dn.d #n.d	*	e' d'	•	d d	d d	d	d	d	d	d d	-	-	•	NDT(bitnofd) → Z 1 → bitnofd	Set Z with state of specified bit in d then set the bit in d
nau		address ²		U	-		U	u	-		u	u	10 V		8		
BSR BTST			*	-		-			-	-	-	-	-			$PC \rightarrow -(SP); address \rightarrow PC$	Branch to subroutine (8 or 16-bit ± offset)
2121		Dn,d #n,d		e' d'	8 7 8	b	d d	d	d d	b	b	b	d	b	070 1.200	NDT(bit Dn of d) \rightarrow Z	Set Z with state of specified bit in d
			-*000	-		d	-	d		d	d	d	-	d	8	NOT(bit #n of d) $\rightarrow Z$	Leave the bit in d unchanged
CHK		s,Dn	-0100	B	-	8	S	S	8	8	S	S	S	8	8	if Dn <d dn="" or="">s then TRAP</d>	Compare Dn with D and upper bound [s]
		d	-0100	d	-	d	d	d	d	d	d	d	-	-	-	D → d	Clear destination to zero
CMP ⁴		s,Dn	_****	B	s ⁴	8	S	8	8	8	8	8	S	8	s ⁴	set CCR with Dn – s	Compare Dn to source
CMPA ⁴		s,An		S	B	S	5	8	S	8	8	S	8	8	8	set CCR with An - s	Compare An to source
CMPI ⁴		#n,d	_****	d		d	d	d	d	d	d	d		-	S	set CCR with d - #n	Compare destination to #n
CMPM ⁴)Bcc	BWL W	(Ay)+,(Ax)+ Dn,addres ²		-	-	-	<u> </u>		-	376 940	-	-	-			set CCR with (Ax) - (Ay) if cc false then { Dn-1 \rightarrow Dn	Compare (Ax) to (Ay); Increment Ax and Ay Test condition, decrement and branch
																if Dn \leftrightarrow -1 then addr \rightarrow PC }	(16-bit ± offset to address)
DIVS		s,Dn	-***0	B). 	S	S	8	8	S	8	S	8	S	8	±32bit Dn / ±16bit s → ±Dn	Dn= [16-bit remainder, 16-bit quotient]
DIAN		s,Dn	-***0	B	-	8	S	S	8	8	S	S	S	8	8	32bit Dn / 16bit s → Dn	Dn= [16-bit remainder, 16-bit quotient]
EDR ⁴		Dn,d	-**00	e	-	d	d	d	d	d	d	d	-	1920	s ⁴	Dn XDR d \rightarrow d	Logical exclusive DR Dn to destination
EDRI ⁴		#n,d	-**00	d		d	d	d	d	d	d	d	-	-	S	#n XDR d → d	Logical exclusive DR #n to destination
EDRI ⁴		#n,CCR		-	्रम्	-	-	-	-	-	-		-	-	8	#n XOR CCR → CCR	Logical exclusive DR #n to CCR
EDRI ⁴	W	#n,SR		-	-	-	19	-	-	-	-	-	-	-	8	#n XDR SR → SR	Logical exclusive DR #n to SR (Privileged)
XG	L	Rx,Ry		B	В	-	-	-	-	-	-	240)	-	-	-	register $\leftarrow \rightarrow$ 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)
LLEGAL				-		-		-	-	270	-		-		. .	$PC \rightarrow -(SSP); SR \rightarrow -(SSP)$	Generate Illegal Instruction exception
IMP		d		-	-	d			d	d	d	d	d	d	-	Îd → PC	Jump to effective address of destination
ISR		d		-	-	d	5 12	-	d	d	d	d	d	d	-	$PC \rightarrow -(SP); \uparrow d \rightarrow PC$	push PC, jump to subroutine at address d
EA	L	s,An		-	е	s	- 14	-	8	8	8	S	8	8	1940	↑s → An	Load effective address of s to An
.INK		An,#n		-	-	-	-	-	-	-	-	-	-		-	$An \rightarrow -(SP); SP \rightarrow An;$ SP + #n \rightarrow SP	Create local workspace on stack (negative n to allocate space)
.SL	BWL	Dx,Dy	***0*	B		-	4	-		120	-	1.2			1920		Logical shift Dy, Dx bits left/right
.SR	6225	#n.Dy		d	-	-	a .	-	-	-	-	-	-	-	5		Logical shift Dy, #n bits L/R (#n: 1 to 8)
ATTUE 4		d	-**00	-	-	d	d	d	d	d	d	d	-	-	-		Logical shift d 1 bit left/right (.W only)
ADVE 4		s.d		B	s ⁴	B	B	B	B	В	B	В	8	8		s → d	Move data from source to destination
ADVE		s,CCR		8	-	8	S	8	8	8	S	S	S	8		$s \rightarrow CCR$	Move source to Condition Code Register
ADVE		s,SR		S	-	S	S	8	8	8	S	8	8	8	-	$s \rightarrow SR$	Move source to Status Register (Privileged)
ADVE	W	SR,d		d	-	d	d	d	d	d	d	d	-	-	٠	$SR \rightarrow d$	Move Status Register to destination
MOVE	L	USP,An An,USP		-	d s	-	-	2	-	-	-	-	-	-	-	$\begin{array}{c} \text{USP} \rightarrow \text{An} \\ \text{An} \rightarrow \text{USP} \end{array}$	Move User Stack Pointer to An (Privileged) Move An to User Stack Pointer (Privileged)
	BWL	s,d	XNZVC	De	-	(10)	(Ап)+	-(An)	(i An)	(i,An,Rn)	ahe W	ahel	(iPC)	(i,PC,Rn)	#n		

Lincode	COMPUTER AFCHITECTURE - EPITA - 53 - 2020/2021 de Size Uperand CCK Effective Address s=source, d=destination, e=either, i=displacement Uperation Uescription																	
Opeous	BWL	s,d	XNZVC		An		(An)+			(i,An,Rn)						оры дайл	Deatription	
MOVEA ⁴		s,An		S	B	S	S	S	S	S	S	S	S	S	-	s → An	Move source to An (MDVE s,An use MDVEA)	
MOVEM ⁴		Rn-Rn,d		-	-	d	-	d	d	d	d	d	-	-	-	Registers $\rightarrow d$	Move specified registers to/from memory	
MUTCH	WL.	s,Rn-Rn				S	S	<u> </u>	8	S	S	S	8	8	_	$s \rightarrow \text{Registers}$	(.W source is sign-extended to .L for Rn)	
MOVEP	WI	Dn,(i,An)		s	-	-	-	-	d	-	-	-	-	-	_	$Dn \rightarrow (i,An)(i+2,An)(i+4,A.$ Move Dn to/from alternate memor		
MUTLI	m	(i,An),Dn		d	-	_	-	_	5	-	-	-	С — — — — — — — — — — — — — — — — — — —	-	-	$(i,An) \rightarrow Dn(i+2,An)(i+4,A.)$	(Access only even or odd addresses)	
MOVEQ ⁴	1	#n,Dn	-**00	d	-	-	-	-	-	-	-	-	-	-	S	#n → Dn	Move sign extended 8-bit #n to On	
MULS	W	s,Dn	-**00	B	-	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	B		S	s	S	8	8	s	S	8	S	8	16bit s * 16bit Dn \rightarrow Dn	Multiply unsig'd 16-bit; result: unsig'd 32-bit	
NBCD	B	d	*U*U*	d		d	d	d	d	d	d	d	-	-	a -	$D - d_0 - X \rightarrow d$	Negate BCD with eXtend, BCD result	
NEG		d	****	d		d	d	d	d	d	d	d	-	-		0-d→d	Negate destination (2's complement)	
NEGX		d	****	d	-	d	d	d	d	d	d	d		-		0-d-X→d	Negate destination with extend	
NDP	DWL	u		u	-	u	u	u	- U		u	u	2		-	None	No operation occurs	
NOT	BWL	d	-**00	d	-	d	d	d	d	d	d	d	-		-	NDT(d) → d	Logical NDT destination (I's complement)	
DR ⁴		s,Dn	-**00	U B	-								-	-	s ⁴	s DR Dn \rightarrow Dn	Logical DR	
пк	DAAL	s,un Dn,d	00		성류의 가능자	s d	s d	s d	s d	s d	s d	s d	S	5	S	Dn DR d \rightarrow d	(ORI is used when source is #n)	
DRI ⁴	BWL	#n.d	-**00	e d	-	d	d	d	d	d	d	d	-	-		un un a → a #n DR d → d	Logical DR #n to destination	
DRI ⁴	B	#n,0 #n,CCR		0	-	-	-	-	-		-		-	-		#n DR CCR \rightarrow CCR	Logical DR #n to CCR	
DRI ⁴	_			-	ः	-		-	-		-			38	-			
	W	#n,SR		-	-			-	-	(1 7 1)		-	. 5	35 7 3	8	$\#_n \text{ DR } SR \rightarrow SR$	Logical DR #n to SR (Privileged)	
PEA	L	8		-	-	S	-	-	8	8	8	8	8	8	-	$\uparrow_{s} \rightarrow -(SP)$	Push effective address of s onto stack	
RESET	-		-**0*	-	-	-	-	-	-	-	-	-	-	14	-	Assert RESET Line	Issue a hardware RESET (Privileged)	
ROL	RML	Dx.Dy	-**0*	B	-	-	(H)	-	-		-	-				: ∢∢	Rotate Dy, Dx bits left/right (without X)	
RDR		#n.Dy		d	-	1	-	1	1	-	1	-	5	-	S		Rotate Dy, #n bits left/right (#n: 1 to 8)	
0.01/1	W	d	***0*	-	-	d	d	d	d	d	d	d	. ×		(4)		Rotate d 1-bit left/right (.W only)	
ROXL	BWL	Dx.Dy		B	((1 1))	-	2.5	-	-	-	-	-		() - ()	((11 3)) 11111		Rotate Dy, Dx bits L/R, X used then updated	
RDXR	w	#n.Dy		d	-	-	1	1	1	-	5	-	5	-	8		Rotate Dy, #n bits left/right (#n: 1 to 8)	
RTE	W	d		-	-	d	d	d	d	d	d	d	-	-	-		Rotate destination 1-bit left/right (.W only)	
				-		-	-			-			-		3.#K)	$(SP) + \rightarrow SR; (SP) + \rightarrow PC$	Return from exception (Privileged)	
RTR				-	-	-	-	-	-		-	-	-	-	•	$(SP)^{+} \rightarrow CCR, (SP)^{+} \rightarrow PC$	Return from subroutine and restore CCR	
RTS		0.0	*U*U*	-	-	-	-	-	-	-	-	-	-	- 12	-	(SP)+ → PC	Return from subroutine	
SBCD	B	Dy,Dx	10101	B	. . .	-	1.2		-	-	-	-				$Dx_{10} - Dy_{10} - X \rightarrow Dx_{10}$	Subtract BCD source and eXtend bit from	
		-(Ay),-(Ax)		-	-	-	1	B	-	-	-	-)):=()) ()::::::::::::::::::::::::::::::	$-(Ax)_{10}(Ay)_{10} - X \rightarrow -(Ax)_{10}$	destination, BCD result	
Scc	B	d		d	-	d	d	d	d	d	d	d	-	-	-	If cc is true then I's \rightarrow d	If cc true then d.B = 11111111	
0700	_					_		_					-		-	else D's \rightarrow d	else d.B = 00000000	
STOP	marr	#n		-	-	-		-	-	-		-	-	-		$\#n \rightarrow SR; STOP$	Move #n to SR, stop processor (Privileged)	
SUB ⁴	BWL	s,Dn	****	B	S 14	S	S	S	8	S	S	S	8	S	s ⁴	Dn - s → Dn	Subtract binary (SUBI or SUBQ used when	
DUD A		Dn,d		B	ď	d	d	d	d	d	d	d	-	1526	-	d - Dn → d	source is #n. Prevent SUBQ with #n.L)	
SUBA ⁴	WL	s,An		S	8	S	S	S	8	S	8	S	S	S	8	An - s → An	Subtract address (.W sign-extended to .L)	
SUBI ⁴	BWL	#n,d	****	d	-	d	d	d	d	d	d	d		(-)				
SUBQ ⁴	BWL	#n,d	*****	d	d	d	d	d	d	d	d	d		873	8			
SUBX	BWL	Dy.Dx	****	B	-	-	25	-	-	-	-	-	-	-	-	$Dx - Dy - X \rightarrow Dx$ Subtract source and eXtend bit fro		
		-(Ay),-(Ax)		-	-	-	-	B	-	-				-		$-(Ax)(Ay) - X \rightarrow -(Ax)$ destination		
SWAP		Dn	-**00		-	-		-	-	-	-	-	-	-		$bits[31:16] \leftrightarrow bits[15:0]$	Exchange the 16-bit halves of Dn	
TAS	B	d	-**00	-	•	d	d	d	d	d	d	d	-	-	•	test $d \rightarrow CCR$; $1 \rightarrow bit7$ of d N and Z set to reflect d, bit7 of d set t		
TRAP		#n		-	-	-	-	-	-	-	-	-	-	-	8	a second s		
						-										(vector table entry) \rightarrow PC	(#n range: 0 to 15)	
TRAPV				-	-	- 7	5		-	17	-	-	5			If V then TRAP #7	If overflow, execute an Overflow TRAP	
TST	BWL		-**00	d	-	d	d	d	d	d	d	d	2		12	test d \rightarrow CCR	N and Z set to reflect destination	
UNLK		An		-	d	-	-	-	-		-	•	-	-	-	An \rightarrow SP: (SP)+ \rightarrow 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			

CC	Condition	Test	CC	Condition	Test		
T	true	1	VC	overflow clear	١V		
F	false	0	VS	overflow set	٧		
HI	higher than	!(C + Z)	PL	plus	IN		
LS"	lower or same	C + Z	MI	minus	N		
HS", CC°	higher or same	10	GE	greater or equal	!(N ⊕ V)		
LO", CS*	lower than	C	LT	less than	$(N \oplus V)$		
NE	not equal	12	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)
- On Data register (8/16/32-bit, n=0-7)
- Rn any data or address register
- Source, **d** Destination S
- Either source or destination B
- Immediate data, i Displacement #n BCD Binary Coded Decimal
- Effective address
- 1
- Long only; all others are byte only 2
 - Assembler calculates offset
 - Branch sizes: .B or .S -128 to +127 bytes, .W or .L -32768 to +32767 bytes
 - Assembler automatically uses A, I, Q or M form if possible. Use #n.L to prevent Quick optimization

SSP Supervisor Stack Pointer (32-bit)

SP Active Stack Pointer (same as A7)

CCR Condition Code Register (lower 8-bits of SR)

N negative, Z zero, V overflow, C carry, X extend

- not affected, O cleared, 1 set, U undefined

* set according to operation's result, \equiv set directly

USP User Stack Pointer (32-bit)

PC Program Counter (24-bit)

SR Status Register (16-bit)

Distributed under the GNU general public use license.

3

4

Last name: Group: 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 #5000,-4(A2)		
MOVE.L \$500A,-(A2)		
MOVE.B 1(A0),-7(A1,D1.W)		
MOVE.L -12(A2),\$13(A1,D2.W)		

Exercise 2

Operation	Size (bits)	Result (hexadecimal)	Ν	Z	V	С
\$42 + \$2A	8					
\$7FF0 + \$11	16					
\$FFFFFF0 + \$11	32					

Exercise 3

Values of registers after the execution of the program. Use the 32-bit hexadecimal representation.						
D1 = \$	D3 = \$					
D 2 = \$	D 4 = \$					

Exercise 4

ToAscii