Your Name: _____ netid:_____ Group #:

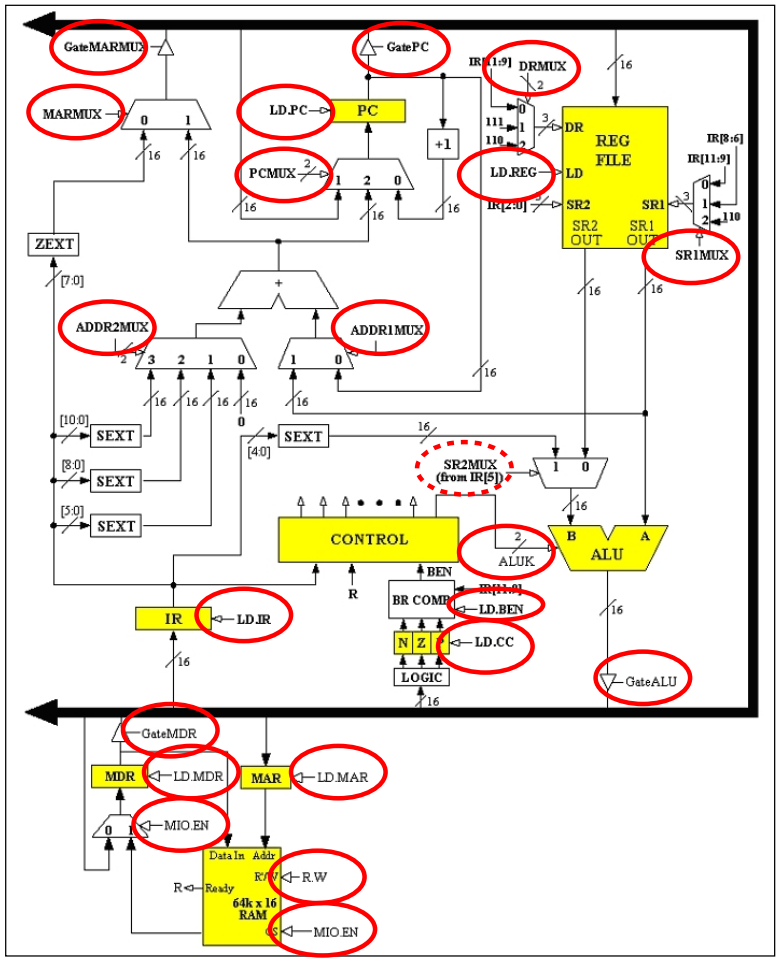
Name: _____ netid:_____

Name: _____ netid:_____

Name: _____ netid:_____

ECE 120 Worksheet 14: LC-3 datapath control

The figure below shows the LC-3 datapath and all the control signals necessary to control it. At each state of the LC-3 FSM, these signals are configured to enable a particular RTL statement to be carried out by the datapath. In lecture, we attempted to show how the datapath needs to be configured to implement the states for the *fetch* phase of the instruction cycle. In this worksheet, you will configure the datapath for the *execute* phase states for some of the LC-3 instructions.



Feel free to detach and keep the last 3 pages of this booklet.

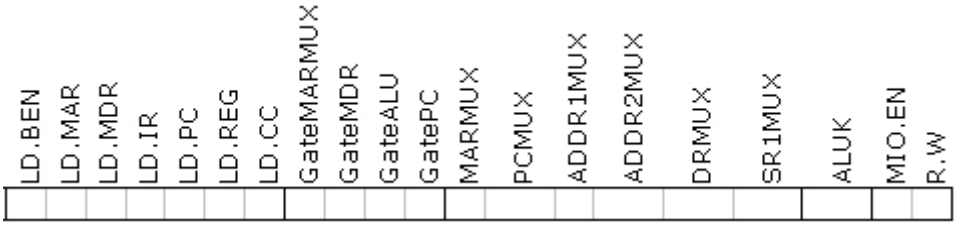
1. LC-3 control signals

Fill in the table below by specifying control bits for the states listed in the table. You may use don't cares where appropriate. The states are listed top-to-bottom, left-to-right as they appear in the LC-3 state diagram. Consult with the LC-3 FSM and datapath attached to this worksheet (you can detach and keep the last 3 pages). States 18 and 32 are done for you as an example.

STATE	RTL	LD.BEN	LD.MAR	LD.MDR	LD.IR	LD.PC	LD.REG	LD.CC	Gate.MARMUX	Gate.MDR	Gate.ALU	Gate.PC	MARMUX	PCMUX (2 BITS)	ADDR1MUX	ADDR2MUX (2 BITS)	DRMUX (2 BITS)	SR1MUX (2 BITS)	ALUK (2 BITS)	MIO.EN	R.W
18	MAR ← PC, PC← PC+1	0	1	0	0	1	0	0	0	0	0	1	x	00	x	xx	xx	xx	xx	0	x
32	BEN ← nN+zZ+pP	1	0	0	0	0	0	0	0	0	0	0	x	xx	x	xx	xx	xx	xx	0	x
5																					
6																					
25																					
27																					

2. LC-3 control words

In the previous problem you noticed that each RTL statement requires configuring 25 LC-3 datapath control signals. These 25 control signals can be packed together as a single 25-digit binary word, or **control word**, assuming some fixed order, e.g.,



For example, control word for the RTL statement implementing the execute phase of ADD instruction is

00000110010xxxxxx0001000x

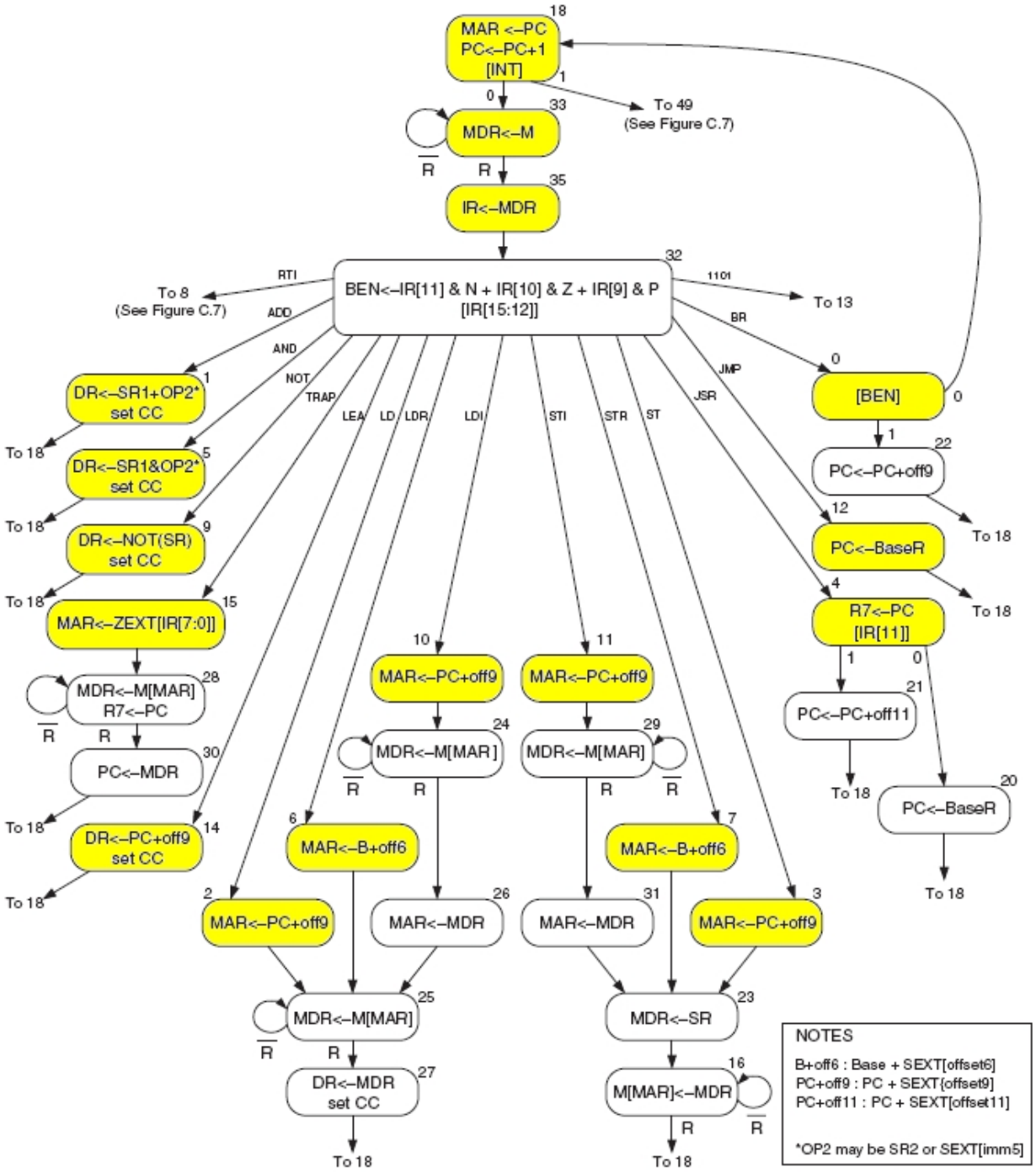
We can add three more 0 bits to the front of this word (so that it is 28-bit long) and convert it into a 7-digit hexadecimal number, replacing all don't cares with 0s:

000**0 0000 1100 1000 0000 0001 0000**$_2$ = x00C8010 <- control word for state 1.

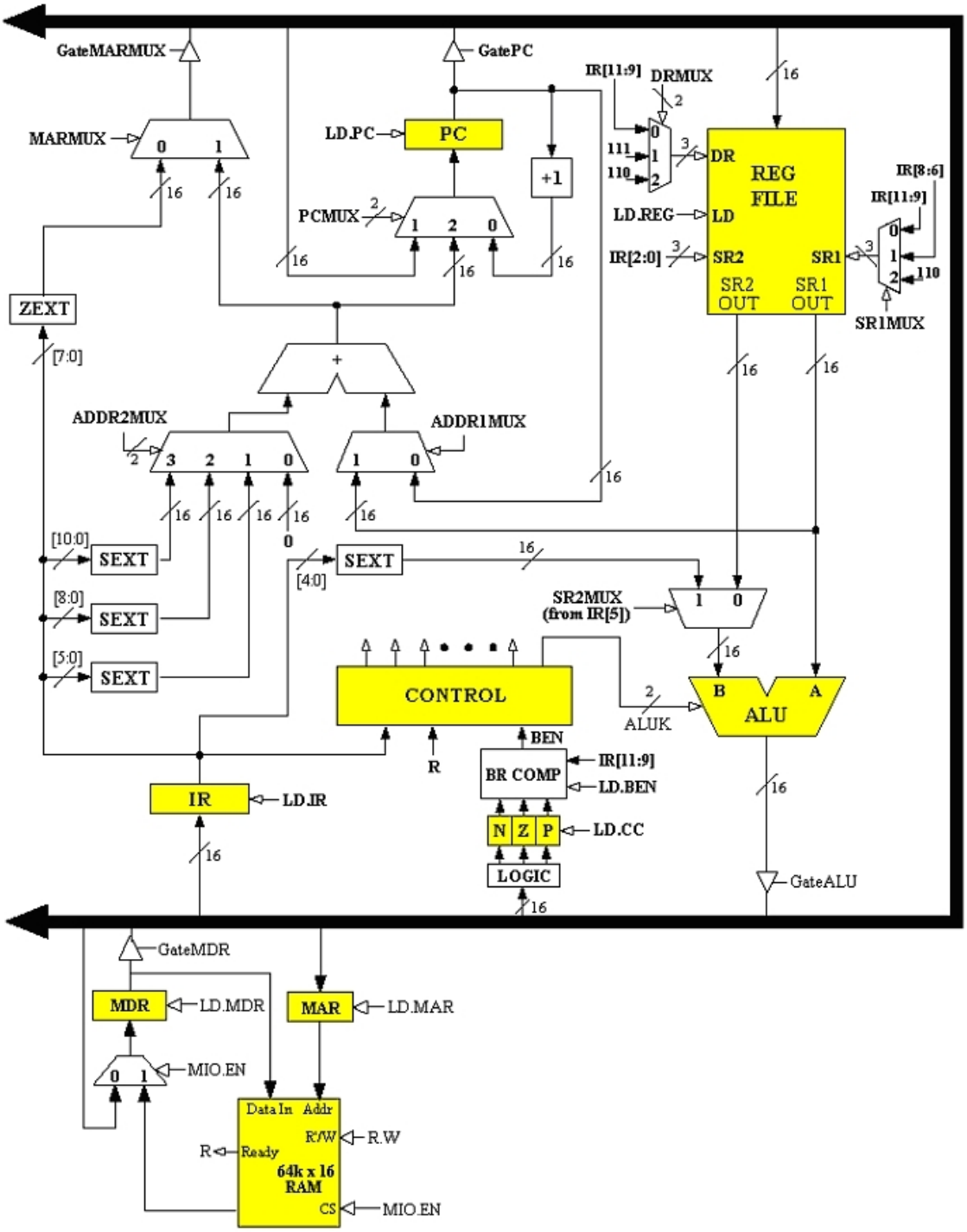
Based on your table from the previous part, write the control words for the following states:

State	Control word in binary (25 bits)	Control word in hexadecimal
18	0 1001 0000 01**0**0 0**000 0000 00**0**0**	**x0504000**
32	1 0000 0000 00**00 0000 0000 00**0**0**	**x1000000**
5		
6		
25		
27		

LC-3 FSM


MAR <–PC
PC<–PC+1
[INT]
18
1
0
To 49
(See Figure C.7)
MDR<–M
33
R
R
IR<–MDR
35
BEN<–IR[11] & N + IR[10] & Z + IR[9] & P
[IR[15:12]]
32
RTI
To 8
(See Figure C.7)
ADD
AND
NOT
TRAP
LEA
LD
LDR
LDI
STI
STR
ST
JSR
JMP
BR
1101
To 13
DR<–SR1+OP2*
set CC
1
To 18
DR<–SR1&OP2*
set CC
5
To 18
DR<–NOT(SR)
set CC
9
To 18
MAR<–ZEXT[IR[7:0]]
15
MDR<–M[MAR]
R7<–PC
28
PC<–MDR
30
To 18
DR<–PC+off9
set CC
14
To 18
MAR<–PC+off9
2
MAR<–B+off6
6
MAR<–PC+off9
10
MDR<–M[MAR]
24
MAR<–MDR
26
MDR<–M[MAR]
25
DR<–MDR
set CC
27
To 18
MAR<–PC+off9
11
MDR<–M[MAR]
29
MAR<–B+off6
7
MAR<–MDR
31
MAR<–PC+off9
3
MDR<–SR
23
M[MAR]<–MDR
16
To 18
[BEN]
0
0
1
PC<–PC+off9
22
To 18
PC<–BaseR
12
To 18
R7<–PC
[IR[11]]
4
1
0
PC<–PC+off11
21
To 18
PC<–BaseR
20
To 18
NOTES
B+off6 : Base + SEXT[offset6]
PC+off9 : PC + SEXT{offset9]
PC+off11 : PC + SEXT[offset11]
*OP2 may be SR2 or SEXT[imm5]

LC-3 datapath



LC-3 control signals

Signal	Description
LD.MAR	= 1, MAR is loaded
LD.MDR	= 1, MDR is loaded
LD.IR	= 1, IR is loaded
LD.PC	= 1, PC is loaded
LD.REG	= 1, register file is loaded
LD.BEN	= 1, updates Branch Enable (BEN) bit

Signal	Description
LD.CC	= 1, updates status bits from system bus
GateMARMUX	= 1, MARMUX output is put onto system bus
GateMDR	= 1, MDR contents are put onto system bus
GateALU	= 1, ALU output is put onto system bus
GatePC	= 1, PC contents are put onto system bus

MARMUX
- = 0, chooses ZEXT IR[7:0]
- = 1, chooses address adder output

ADDR1MUX
- = 0, chooses PC
- = 1, chooses reg file SR1OUT

ADDR2MUX
- = 00, chooses "0...00"
- = 01, chooses SEXT IR[5:0]
- = 10, chooses SEXT IR[8:0]
- = 11, chooses SEXT IR[10:0]

PCMUX
- = 00, chooses PC + 1
- = 01, chooses system bus
- = 10. chooses address adder output

SR1MUX
- = 00, chooses IR[11:9]
- = 01, chooses IR[8:6]
- = 10, chooses "110"

MIO.EN
- = 1, Enables memory, chooses memory output for MDR input
- = 0, Disables memory, chooses system bus for MDR input

R.W
- = 1, M[MAR]<-MDR when MIO.EN = 1
- = 0, MDR<-M[MAR] when MIO.EN = 1

ALUK
- = 00, ADD
- = 01, AND
- = 10, NOT A
- = 11, PASS A

DRMUX
- = 00, chooses IR[11:9]
- = 01, chooses "111"
- = 10, chooses "110"

LC-3 Instructions

NOTES: RTL corresponds to execution (after fetch!); JSRR not shown

ADD	0001	DR	SR1	0	00	SR2	ADD DR, SR1, SR2		DR ← SR1 + SR2, Setcc
ADD	0001	DR	SR1	1	imm5		ADD DR, SR1, *imm5*		DR ← SR1 + SEXT(imm5), Setcc
AND	0101	DR	SR1	0	00	SR2	AND DR, SR1, SR2		DR ← SR1 AND SR2, Setcc
AND	0101	DR	SR1	1	imm5		AND DR, SR1, *imm5*		DR ← SR1 AND SEXT(imm5), Setcc
BR	0000	n	z	p	PCoffset9		BR{nzp} *PCoffset9*		((n AND N) OR (z AND Z) OR (p AND P)): PC ← PC + SEXT(PCoffset9)
JMP	1100	000	BaseR	000000			JMP BaseR		PC ← BaseR
JSR	0100	1	PCoffset11				JSR *PCoffset11*		R7 ← PC, PC ← PC + SEXT(PCoffset11)
TRAP	1111	0000	trapvect8				TRAP *trapvect8*		R7 ← PC, PC ← M[ZEXT(trapvect8)]
LD	0010	DR	PCoffset9				LD DR, *PCoffset9*		DR ← M[PC + SEXT(PCoffset9)], Setcc
LDI	1010	DR	PCoffset9				LDI DR, *PCoffset9*		DR ← M[M[PC + SEXT(PCoffset9)]], Setcc
LDR	0110	DR	BaseR	offset6			LDR DR, BaseR, *offset6*		DR ← M[BaseR + SEXT(offset6)], Setcc
LEA	1110	DR	PCoffset9				LEA DR, *PCoffset9*		DR ← PC + SEXT(PCoffset9), Setcc
NOT	1001	DR	SR	111111			NOT DR, SR		DR ← NOT SR, Setcc
ST	0011	SR	PCoffset9				ST SR, *PCoffset9*		M[PC + SEXT(PCoffset9)] ← SR
STI	1011	SR	PCoffset9				STI SR, *PCoffset9*		M[M[PC + SEXT(PCoffset9)]] ← SR
STR	0111	SR	BaseR	offset6			STR SR, BaseR, *offset6*		M[BaseR + SEXT(offset6)] ← SR