Quinn Roemer

Professor Chris Grove

CSC 137

06 April 2020

Homework #4

Question 1:

For each entry in the F1 table (Table 7-1), translate it into the actual signal names. Include the multiplexor if needed.

F1	Microoperation	Symbol	Signal Names
000	None	NOP	-
001	AC <- AC + DR	ADD	Ld(AC), Add
010	AC <- 0	CLRAC	Clr(AC)
011	AC <- AC + 1	INCAC	Inr(AC)
100	AC <- DR	DRTAC	LD(AC), DRTAC
101	AR <- DR(0-10)	DRTAR	LD(AR), Mux(1)
110	AR <- PC	PCTAR	LD(AR), Mux(0)
111	M[AR] <- DR	WRITE	Write

7.7:

Using the mapping procedure of Fig. 7.3, map out the microcode addresses for these operation codes: 0010, 1011, 1111.

Opcode	*Microinstruction Address*
0010	0001000
1011	0101100
1111	0111100

7.11:

With table 7.1, convert these microoperation symbols to microoperation and binary equivalents.

Microoperation	*F1*	*F2*	*F3*	*Microoperation Symbols*
AC <- AC + 1, DR <- DR + 1	011	110	000	INCAC, INCDR, NOP
PC <- PC +1, DR <- M[AR]	000	100	101	NOP, READ, INCPC
DR <- AC, AC <- DR	100	101	000	DRTAC, ACTDR, NOP

7.12:

With table 7.1, convert these microoperation symbols to microoperation and binary equivalents

Microoperation Symbols	*F1*	*F2*	*F3*	*Microoperation*
READ, INCPC	000	100	101	DR <- M[AR], PC <- PC +1
ACTDR, DRTAC	100	101	000	AC <- DR, DR <- AC
ARTPC, DRTAC, WRITE	100 & 111	000	110	AC <- DR, M[AR] <- DR, PC <- AR

NOTE: Last microoperation is impossible since it requires two F1 control signals.

7.15:

With the following microcode (Sec. 7.3)

a. Translate to symbols (Table 7.2).

Address & Microcode	*F1*	*F2*	*F3*	*CD*	*BR*	*AD*
60 010 000 010 00 00 1000011	CLRAC	NOP	COM	U	JMP	INDRCT
61 111 100 000 01 01 1000000	WRITE	READ	NOP	I	CALL	FETCH
62 001 001 000 10 10 0111111	ADD	SUB	NOP	S	RET	NEXT
63 101 110 000 11 11 0111100	DRTAR	INCDR	NOP	Z	MAP	60

b. List the things logically wrong in the code.

1. In the first instruction you unconditionally branch to the INDRCT subroutine without saving a return. Therefore, the return at the end of the INDRCT subroutine would return to the last saved address breaking program flow.
2. In the second instruction you are reading and writing at the same time. This is impossible due to the way in which the registers are connecting in figure 7.4. Also, a call to the fetch subroutine saves a return address that will not be used. As fetch increments PC without returning.
3. In the third instruction it is impossible to perform an ADD and SUB command at the same time since these are conflicting ALU operations. In addition, performing a RET does not rely on the value of S.
4. In the fourth instruction, the MAP command is performed without reliance on Z.

7.16:

Add the following new commands to Table 7.2.

AND 0100 AC <- AC ^ M[AR] - Bitwise AND

	ORG 16			
AND	NOP	I	CALL	INDRCT
	READ	U	JMP	NEXT
	AND	U	JMP	FETCH

SUB 0101 AC <- AC – M[AR] – Subtraction

	ORG 20			
SUB	NOP	I	CALL	INDRCT
	READ	U	JMP	NEXT
	SUB	U	JMP	FETCH

ADM 0110 M[AR] <- M[AR] + AC – Add to memory

	ORG 24			
SUB	NOP	I	CALL	INDR
	READ	U	JMP	NEXT
	DRTAC, ACTDR	U	JMP	NEXT
	ADD	U	JUMP	EXCHANGE + 2

7.17:

Write the microoperation symbols for the CPU instruction ISC of Chapter 5 (Table 5.4). Note DR = 0 status condition is not available in the CD field in Sec 7.3. However, you can exchange AC and DR and check if AC = 0.

ISZ	NOP	I	CALL	INDRCT
	READ	U	JMP	NEXT
	INCDR	U	JMP	NEXT
	DRTAC, ACTDR	U	JMP	NEXT
	DRTAC, ACTDR	Z	JMP	ISZERO
	WRITE	U	JMP	FETCH
ISZERO	WRITE, INCPC	U	JMP	FETCH

7.18:

Write the microoperation symbols for the BSA instruction like the previous problem.

BSA	NOP	I	CALL	INDRCT
	PCTDR, ARTPC	U	JMP	NEXT
	WRITE, INCPC	U	JMP	FETCH