UNIVERSITY OF CALIFORNIA
Department of Electrical Engineering and Computer Sciences
CS150 Fall 2001

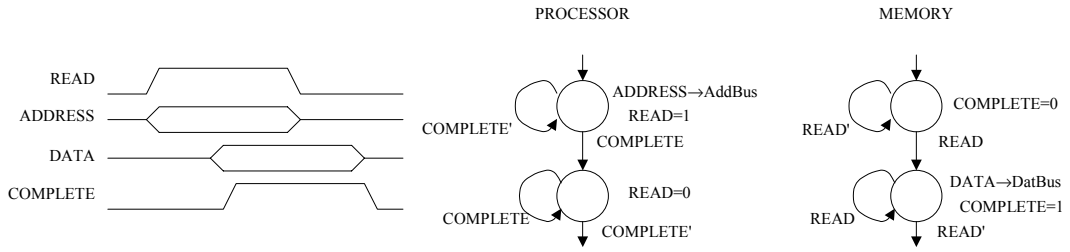
Prof. Subramanian

Homework #8

Due: Friday, November 30, 2001

1) Problem 11.2 on the book.

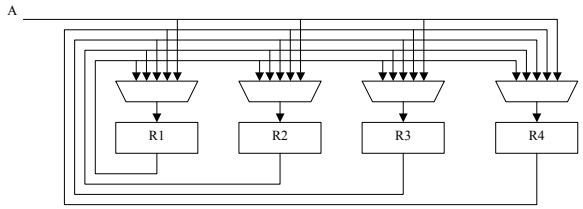
Solution



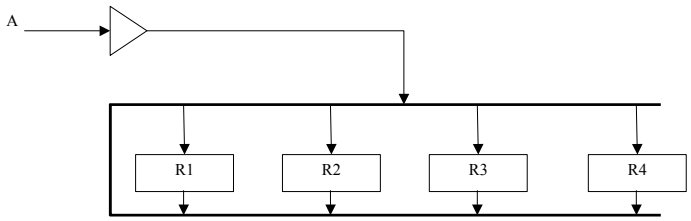
2) Given four registers (R1, R2, R3 and R4), and an additional signal A, you are required to design the interconnection between them so that the resulting datapath can support the operations described in the following cases:
   a) transfer the value of any register or A into any register, and being able to do several transfers at the same time.
   b) transfer the value of any register or A into any register, but only one transfer at a time.
   c) divide the registers into two groups: group 1 contains R1 and R2, and group 2 contains R3 and R4; you should be able to transfer one register of one group or A into one or both registers of the other group, and being able to do two inter-group transfers at the same time.

Solution

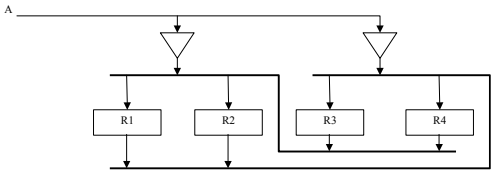
a) Point-to-point connection:



b) Bus:



c) Two buses:



3) Given the datapath shown in section 9, page 1-20 of the lecture notes, describe in register transfer notation the execution of the following instructions:
   a) rt = 55h (Hint: the constant 55h is located in the memory right after the opcode)
   b) rt = mem(10h) (Hint: the constant 10h is located in the memory right after the opcode)

Solution

a) Instruction Fetch:
   mabus←PC (PCmaEN)
   memread (mr)
   IR←memory (IRld)
   op←add (srcA, srcB1, op)
   PC←ALUout (PCld)

Instruction Decode:
   IR to controller

Instruction Execution:
   mabus←PC (PCmaEN)
   memread (mr)
   MBR←memory (MBRld)
   op←add (srcA, srcB1, op)
   PC←ALUout (PCld)
   rt←MBR (regWrite, wrDataSel, wrRegSel)

b) Instruction Fetch:
   (as in the previous part)

Instruction Decode:
   (as in the previous part)

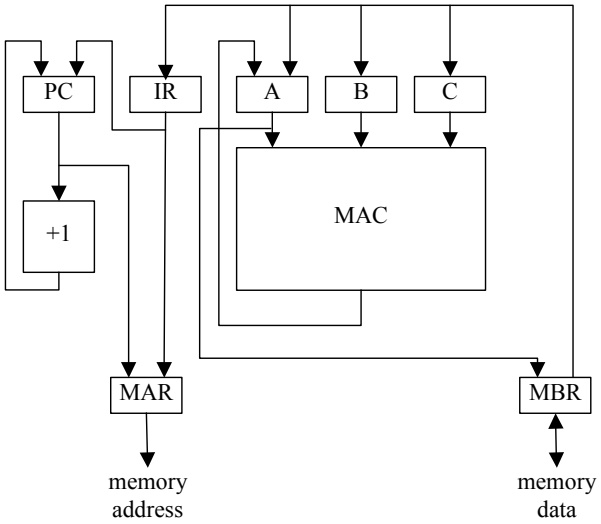
Instruction Execution:
   Load the constant:
      mabus←PC (PCmaEN)
      memread (mr)
      MBR←memory (MBRld)
      op←add (srcA, srcB1, op)
      PC←ALUout (PCld)
   Send the constant to the address bus and read the memory contents:
      This operation is not possible with the given datapath, because there is no connection between the MBR output and the memory address bus; the only possibility is to pass the value from the MBR to the register bank, and from there thru the ALU to the memory address bus. Since the register rt will be overwritten with the new value, we can use it to store temporarily the constant.
      rt←MBR(regWrite, wrDataSel, wrRegSel)
      op←pass (srcA, op)
      mabus←ALUout (ALUmaEN)
      memread (mr)
      MBR←memory (MBRld)
      rt←MBR (regWrite, wrDataSel, wrRegSel)

4) The figure shows the datapath of a simple Digital Signal Processor. The MAC unit performs the multiply-accumulate operation (A+B*C). The registers have a load control signal (PCld, IRld, Ald, Bld, Cld, MARld) and those with two inputs have a select signal (PCsel, Asel, MARsel and MBRsel). Additionally, MBR can enable a buffer to let the data flow from the memory to the registers by using the signal MBRen.

a) Describe in register transfer notation how would you implement the instruction A=n, where n is a constant value. (Include the opcode fetch).

b) Which of the following instructions you can't implement with this datapath?
   i) B=mem(n)
   ii) A=A+B*C
   iii) B=C
   iv) jump n
   v) A=B+C
   vi) jump A

c) Give a sequence of valid instructions to perform $Y=X^2$, where X and Y are variables located in mem(10h) and mem(11h) respectively.



Solution

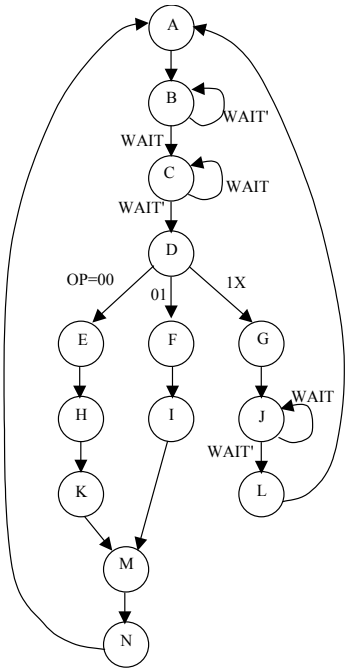
a)

Fetch 1:	PC→MAR (MARsel, MARld)
	PC+1→PC (PCsel, PCld)
Fetch 2:	mem→IR (MBRen, IRld, memread)
Decode:	IR goes to controller
Execute 1:	PC→MAR (MARsel, MARld)
	PC+1→PC (PCsel, PCld)
Execute 2:	mem→A (MBRen, Ald, memread)

b)
i) yes
ii) yes
iii) no
iv) yes
v) no
vi) no

c)
A=0
B=mem(10h)
C=mem(10h)
A=A+B*C
mem(11h)=A

5) Given the state diagram of the figure, design an implementation of the FSM using the jump counter method. Your design should be complete, including state assignment, minimized Boolean expressions for Clear, Load and Count, jump state logic (if you use a ROM, give the contents) and schematic.



Solution

State assignment:

Q3..Q0	state	Q3..Q0	state
0000	A	1000	N
0001	B	1001	F
0010	C	1010	I
0011	D	1011	G
0100	E	1100	J
0101	H	1101	L
0110	K		
0111	M		

CNT = A + B.WAIT + C.WAIT' + E + H + K + M + F + G + J.WAIT'

LD = D + I

CLR = N + L

Q3..Q0	WAIT	CNT	LD	CLR	Q3..Q0	WAIT	CNT	LD	CLR
0000	0	1	0	0	1000	0	0	0	1
	1	1	0	0		1	0	0	1
0001	0	0	0	0	1001	0	1	0	0
	1	1	0	0		1	1	0	0
0010	0	1	0	0	1010	0	0	1	0
	1	0	0	0		1	0	1	0
0011	0	0	1	0	1011	0	1	0	0
	1	0	1	0		1	1	0	0
0100	0	1	0	0	1100	0	1	0	0
	1	1	0	0		1	0	0	0
0101	0	1	0	0	1101	0	0	0	1
	1	1	0	0		1	0	0	1
0110	0	1	0	0	1110	0	X	X	X
	1	1	0	0		1	X	X	X
0111	0	1	0	0	1111	0	X	X	X
	1	1	0	0		1	X	X	X

Using K-maps:

CNT = WAIT'.Q3'.Q0'+WAIT'.Q2.Q0'+WAIT.Q3'.Q1'+Q3'.Q2+Q3.Q2'.Q0

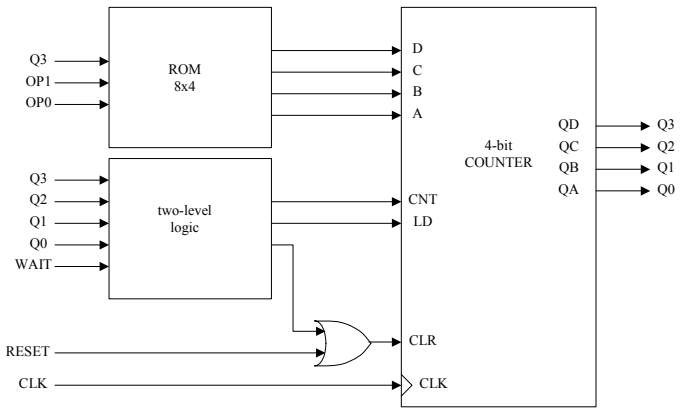
LD = Q3'.Q2'.Q1.Q0+Q3.Q1.Q0'

CLR = Q3.Q2'.Q1'.Q0'+Q3.Q2.Q0

For the jump ROM, we need the inputs OP1 and OP0, and we need to differentiate D (0011) from I (1010), so we select Q3 also as an input. Therefore the contents has to be:

address (Q3,OP1,OP0)	contents
000	0100
001	1001
010	1011
011	1011
100	0111
101	0111
110	0111
111	0111

Finally, the schematic is:



6) Using the datapath of Problem 4, identify all the microoperations. Write the opcode fetch implementation using a horizontal microcode format.

Solution

The microoperations are:

PCld (active high)
PCsel ('0'=PC+1, '1'=IR)
IRld (active high)
Ald (active high)
Asel ('0'=A+B*C, '1'=MBR)
Bld (active high)
Cld (active high)
MARld (active high)
MARsel ('0'=PC, '1'=IR)
MBRld (active high)
MBRsel ('0'=A, '1'=memory)
MBRen (active high)
memread (active high)
memwrite (active high)

The opcode fetch (from problem 4 solution) has two states and can be coded as:

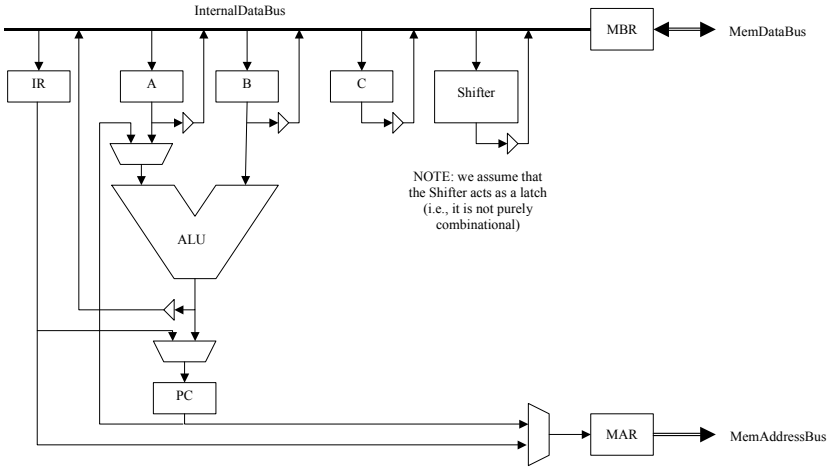
current state	next state	PCld	PCsel	IRld	Ald	Asel	Bld	Cld	MARld	MARsel	MBRld	MBRsel	MBRen	memread	memwrite
1	2	1	0	0	0	0	0	0	1	0	0	0	0	0	0
2	3	0	0	1	0	0	0	0	0	0	0	0	1	1	0

NOTE: for this problem, we don't care about the next state part.

7) A processor has 3 general purpose registers (A,B,C) and is able to perform the following operations: load/store value from/to memory to/from any register, add A+B and store result in any register, subtract A-B and store result in any register, shift right/left any register, jump to any memory location unconditionally and conditionally on zero or sign of the result.
   a) Draw a datapath for this processor (you can use an ALU and a Shifter).
   b) Select a coding for the instructions, grouping the bits in a convenient way.
   c) List the microoperations implied by the datapath.
   d) Write a sequence of microoperations to perform a conditional jump on the sign of the result from the ALU. The jump instruction mnemonic is JP N,label and means: if the sign is negative, jump to the address 'label'.

Solution

a)



b) We use 2 bits to code the type of instruction (data transfer, ALU, Shifter or jump), 2 more bits to distinguish which instruction it is, 2 bits to select a register (if applicable) and finally n bits to select the address or second register (if applicable).

instruction	type (2)	subtype (2)	register (2)	address (n)
ld r,(mem)	00	00	rr	aaa...a
ld (mem),r	00	01	rr	aaa...a
ld r1,r2	00	10	r1r1	r2r2x...x
r=A+B	01	00	rr	xxx...x
r=A-B	01	01	rr	xxx...x
sr r	10	00	rr	xxx...x
sl r	10	01	rr	xxx...x
jp address	11	00	xx	aaa...a
jp z,address	11	01	xx	aaa...a
jp n,address	11	10	xx	aaa...a

c)

A→IBus	(Aen)
IBus→A	(Ald)
B→IBus	(Ben)
IBus→B	(Bld)
C→IBus	(Cen)
IBus→C	(Cld)
Shifter→IBus	(Sen)
IBus→Shifter	(Sld,ShiftR/L)
A+B→IBus	(Asel=A,ALUen, ALUop=sum)
A-B→IBus	(Asel=A,ALUen, ALUop=sub)
PC+1→PC	(Asel=PC,ALUop=inc,PCsel=ALU,PCld)
mem→IBus	(MBRrd)
IBus→mem	(MBRwr)
IBus→IR	(IRld)
IR→PC	(PCsel=IR,PCld)
PC→memadd	(MARsel=PC,MARld)
IR→memadd	(MARsel=IR,MARld)

d) Instruction fetch:

1) PC→memadd
   PC+1→PC
2) mem→IBus
   IBus→IR

Instruction decode:

3) IR goes to the controller
   controller looks at ALUsign
   if ALUsign=0, go back to 1)
   if ALUsign=1, go to 4)

Instruction execution:

4) IR→PC