Chapter 7 – Basic Processing Unit

7.1. The WMFC step is needed to synchronize the operation of the processor and the main memory.

7.2. Data requested in step 1 are fetched during step 2 and loaded into MDR at the end of that clock cycle. Hence, the total time needed is 7 cycles.

7.3. Steps 2 and 5 will take 2 cycles each. Total time = 9 cycles.

7.4. The minimum time required for transferring data from one register to register Z is equal to the propagation delay + setup time
= 0.3 + 2 + 0.2 = 2.5 ns.

7.5. For the organization of Figure 7.1:

(*a*) 1. PC_{out}, MAR_{in}, Read, Select4, Add, Z_{in}
2. Z_{out}, PC_{in}, Y_{in}, WMFC
3. MDR_{out}, IR_{in}
4. PC_{out}, MAR_{in}, Read, Select4, Add, Z_{in}
5. Z_{out}, PC_{in}, Y_{in}
6. $R1_{out}$, Y_{in}, WMFC
7. MDR_{out}, SelectY, Add, Z_{in}
8. Z_{out}, $R1_{in}$, End

(*b*) 1-4. Same as in (*a*)
5. Z_{out}, PC_{in}, WMFC
6. MDR_{out}, MAR_{in}, Read
7. $R1_{out}$, Y_{in}, WMFC
8. MDR_{out}, Add, Z_{in}
9. Z_{out}, $R1_{in}$, End

(*c*) 1-5. Same as in (*b*)
6. MDR_{out}, MAR_{in}, Read, WMFC
7-10. Same as 6-9 in (*b*)

7.6. Many approaches are possible. For example, the three machine instructions implemented by the control sequences in parts *a*, *b*, and *c* can be thought of as one instruction, Add, that has three addressing modes, Immediate (Imm), Absolute (Abs), and Indirect (Ind), respectively. In order to simplify the decoder block, hardware may be added to enable the control step counter to be conditionally loaded with an out-of-sequence number at any time. This provides a ”branching” facility in the control sequence. The three control sequences may now be merged into one, as follows:

1-4. Same as in (*a*)
5. Z_{out}, PC_{in}, If Imm branch to 10

6. WMFC
7. MDR$_{out}$, MAR$_{in}$, Read, If Abs branch to 10
8. WMFC
9. MDR$_{out}$, MAR$_{in}$, Read
10. R1$_{out}$, Y$_{in}$, WMFC
11. MDR$_{out}$, Add, Z$_{in}$
12. Z$_{out}$, R1$_{in}$, End

Depending on the details of hardware timing, steps, 6 and 7 may be combined. Similarly, steps 8 and 9 may be combined.

7.7. Following the timing model of Figure 7.5, steps 2 and 5 take 16 ns each. Hence, the 7-step sequence takes 42 ns to complete, and the processor is idle 28/42 = 67% of the time.

7.8. Use a 4-input multiplexer with the inputs 1, 2, 4, and Y.

7.9. With reference to Figure 6.7, the control sequence needs to generate the Shift right and Add/Noadd (multiplexer control) signals and control the number of additions/subtractions performed. Assume that the hardware is configured such that register Z can perform the function of the accumulator, register TEMP can be used to hold the multiplier and is connected to register Z for shifting as shown. Register Y will be used to hold the multiplicand. Furthermore, the multiplexer at the input of the ALU has three inputs, 0, 4, and Y. To simplify counting, a counter register is available on the bus. It is decremented by a control signal Decrement and it sets an output signal Zero to 1 when it contains zero. A facility to place a constant value on the bus is also available.

After fetching the instruction the control sequence continues as follows:

4. Constant=32, Constant$_{out}$, Counter$_{in}$
5. R1$_{out}$, TEMP$_{in}$
6. R2$_{out}$, Y$_{in}$
7. Z$_{out}$, if TEMP$_0 = 1$ then SelectY else Select0, Add, Z$_{in}$, Decrement
8. Shift, if Zero=0 then Branch 7
9. Z$_{out}$, R2$_{in}$, End

7.10. The control steps are:

1-3. Fetch instruction (as in Figure 7.9)
4. PC$_{out}$, Offset-field-of-IR$_{out}$, Add, If N = 1 then PC$_{in}$, End

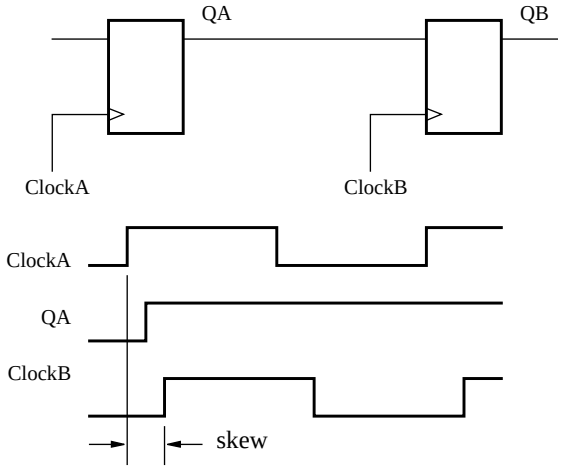
7.11. Let SP be the stack pointer register. The following sequence is for a processor that stores the return address on a stack in the memory.

1-3. Fetch instruction (as in Figure 7.6)
4. SP_{out}, Select4, Subtract, Z_{in}
5. Z_{out}, SP_{in}, MAR_{in}
6. PC_{out}, MDR_{in}, Write, Y_{in}
7. Offset-field-of-IR_{out}, Add, Z_{in}
8. Z_{out}, PC_{in}, End, WMFC

7.12. 1-3. Fetch instruction (as in Figure 7.9)
4. SP_{outB}, Select4, Subtract, SP_{in}, MAR_{in}
5. PC_{out}, R=B, MDR_{in}, Write
6. Offset-field-of-IR_{out}, PC_{out}, Add, PC_{in}, WMFC, End

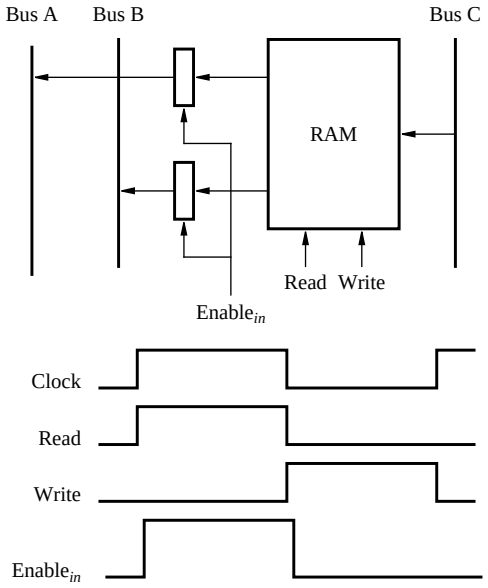
7.13. The latch in Figure A.27 cannot be used to implement a register that can be both the source and the destination of a data transfer operation. For example, it cannot be used to implement register Z in Figure 7.1. It may be used in other registers, provided that hold time requirements are met.

7.14. The presence of a gate at the clock input of a flip-flop introduces clock skew. This means that clock edges do not reach all flip-flops at the same time. For example, consider two flip-flops A and B, with output QA connected to input DB. A clock edge loads new data into A, and the next clock edge transfers these data to B. However, if clock B is delayed, the new data loaded into A may reach B before the clock and be loaded into B one clock period too early.

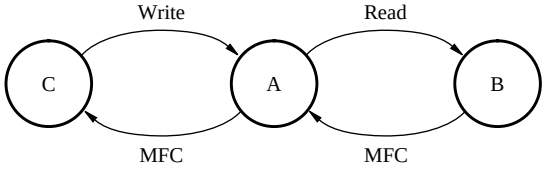


In the absence of clock skew, flip-flop B records a 0 at the first clock edge. However, if Clock B is delayed as shown, the flip-flop records a 1.

7.15. Add a latch similar to that in Figure A.27 at each of the two register file outputs. A read operation is performed in the RAM in the first half of a clock cycle and the latch inputs are enabled at that time. The data read enter the two latches and appear on the two buses immediately. During the second phase of the clock the latch inputs are disabled, locking the data in. Hence, the data read will continue to be available on the buses even if the outputs of the RAM change. The RAM performs a write operation during this phase to record the results of the data transfer.

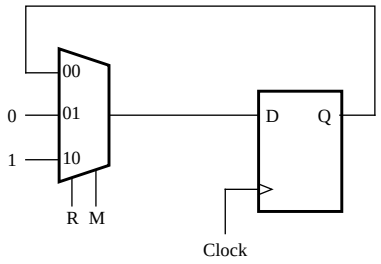


7.16. The step counter advances at the end of a clock period in which Run is equal to 1. With reference to Figure 7.5, Run should be set to 0 during the first clock cycle of step 2 and set to 1 as soon as MFC is received. In general, Run should be set to 0 by WMFC and returned to 1 when MFC is received. To account for the possibility that a memory operation may have been already completed by the time WMFC is issued, Run should be set to 0 only if the requested memory operation is still in progress. A state machine that controls bus operation and generates the run signal is given below.

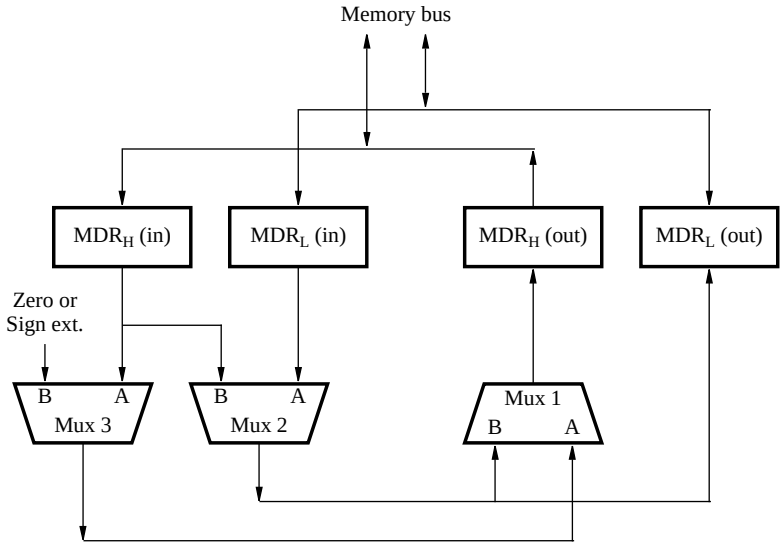


Run = WNFC · (B + C)

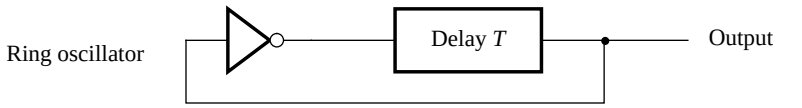
7.17. The following circuit uses a multiplexer arrangement similar to that in Figure 7.3.

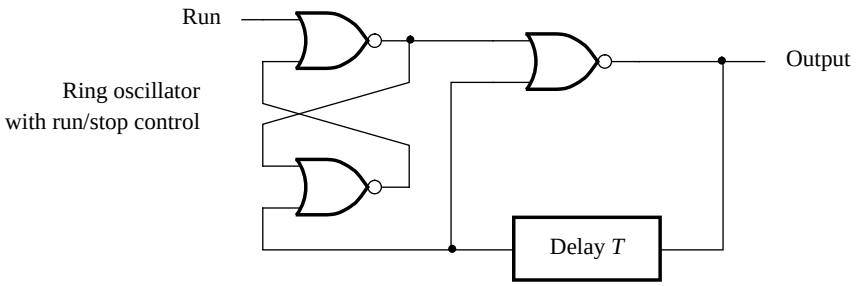


7.18. A possible arrangement is shown below. For clarity, we have assumed that MDR consists of two separate registers for input and output data. Multiplexers Mux-1 and Mux-2 select input B for even and input A for odd byte operations. Mux 3 selects input A for word operations and input B for byte operations. Input B provides either zero extension or sign extension of byte operands. For sign-extension it should be connected to the most-significant bit output of multiplexer Mux-2.

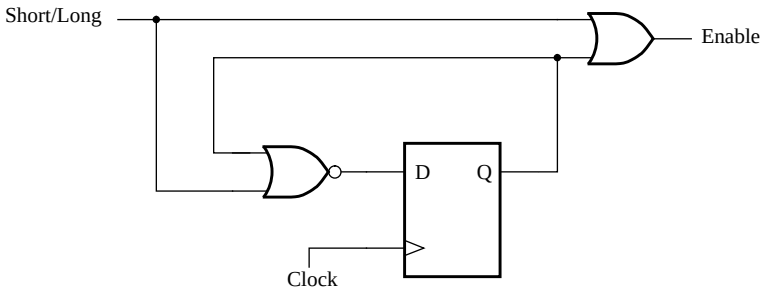


7.19. Use the delay element in a ring oscillator as shown below. The frequency of oscillation is 1/(2T). By adding the control circuit shown, the oscillator will run only while Run is equal to 1. When stopped, its output A is equal to 0. The oscillator will always generate complete output pulses. If Run goes to 0 while A is 1, the latch will not change state until B goes to 1 at the end of the pulse.

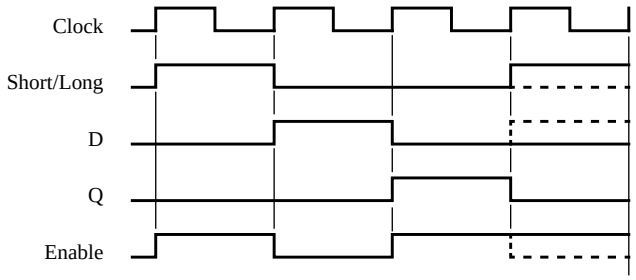




7.20. In the circuit below, Enable is equal to 1 whenever Short/$\overline{\text{Long}}$ is equal to 1, indicating a short pulse. When this line changes to 0, Enable changes to 0 for one clock cycle.



Short/Long	Q	D
0	0	1
0	1	0
1	0	0
1	1	0



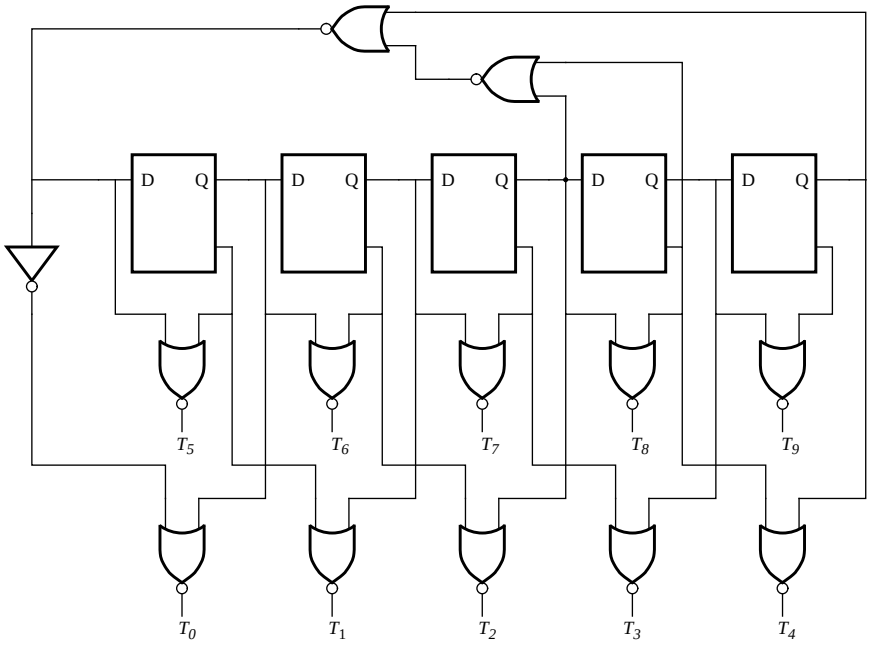
7.21. (*a*) Count sequence is: 0000 1000 1100 1110 1111 0111 0011 0001 0000

(*b*) A 5-bit Johnson counter is shown below, with the outputs Q_1 to Q_5 decoded to generate the signals T_1 to T_{10}. The feed back circuit has been modified to make the counter self-starting. It implements the function

$$D_1 = \overline{Q_5 + \overline{Q_3 + \overline{Q4}}}$$

This circuit detects states that have $Q_3Q_4Q_5$ = 010 and changes the feedback value from 1 to 0. Without this or a similar modification to the feedback circuit, the counter may be stuck in sequences other than the desired one above.

The advantage of a Johnson counter is that there are no glitches in decoding the count value to generate the timing signals.



7.22. We will generate a signal called Store to recirculate data when no external action is required.

$$\begin{aligned}
\text{Store} &= \overline{(\text{ARS} + \text{LSR} + \text{SL} + \text{LLD})} \\
D_{15} &= \text{ASR} \cdot Q_{15} + \text{SL} \cdot Q_{14} + \text{ROR} \cdot \text{Carry} + \text{LD} \cdot D_{15} + \text{Store} \cdot Q_{15} \\
D_1 &= (\text{ASR} + \text{LSR} + \text{ROR}) \cdot Q_2 + \text{SL} \cdot Q_0 + \text{LD} \cdot D_1 + \text{Store} \cdot Q_1 \\
D_0 &= (\text{ASR} + \text{LSR} + \text{ROR}) \cdot Q_1 + \text{LD} \cdot D_0 + + \text{Store} \cdot Q_0
\end{aligned}$$

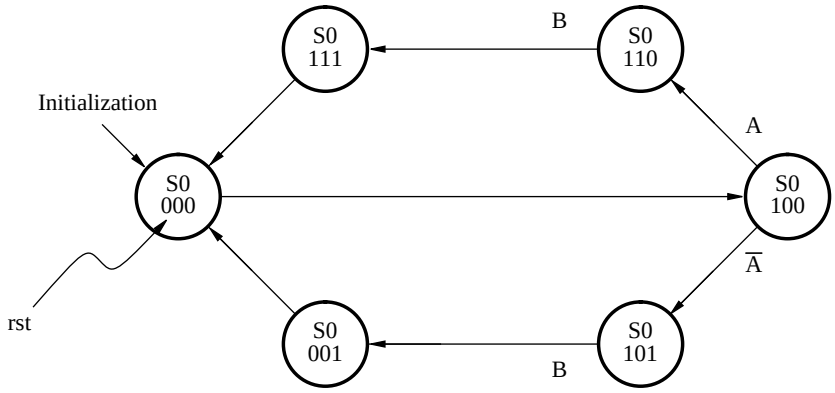
7.23. A state diagram for the required controller is given below. This is a Moore machine. The output values are given inside each state as they are functions of the state only.

Since there are 6 independent states, a minimum of three flip-flops r, s, and t are required for the implementation. A possible state assignment is shown in the diagram. It has been chosen to simplify the generation of the outputs X, Y, and Z, which are given by

$$\mathrm{X} = \mathrm{r} + \mathrm{s} + \mathrm{t} \qquad \mathrm{Y} = \mathrm{s} \qquad \mathrm{Z} = \mathrm{t}$$

Using D flip-flops for implementation of the controller, the required inputs to the flip-flops may be generated as follows

$$\begin{aligned} \mathrm{D(r)} &= \mathrm{s}\,\overline{\mathrm{t}}\,\mathrm{B} + \overline{\mathrm{s}}\,\overline{\mathrm{t}} \\ \mathrm{D(s)} &= \overline{\mathrm{s}}\,\overline{\mathrm{t}}\,\mathrm{A} + \mathrm{s}\,\overline{\mathrm{t}}\,\mathrm{B} \\ \mathrm{D(t)} &= \mathrm{s}\,\overline{\mathrm{t}}\,\mathrm{B} + \overline{\mathrm{s}}\,\overline{\mathrm{t}}\,\overline{\mathrm{A}} + \overline{\mathrm{s}}\,\mathrm{t}\,\mathrm{B} \end{aligned}$$



7.24. Microroutine:

Address (Octal)	**Microinstruction**
000-002	Same as in Figure 7.21
300	μBranch $\{\mu\mathrm{PC} \leftarrow 161$
161	PC_{out}, MAR_{in}, Read, Select4, Add, Z_{in}
162	Z_{out}, PC_{in}, WMFC
163	MDR_{out}, Y_{in}
164	Rsrc_{out}, SelectY, Add, Z_{in}
165	Z_{out}, MAR_{in}, Read
166	μBranch $\{\mu\mathrm{PC} \leftarrow 170$; $\mu\mathrm{PC}_0 \leftarrow \overline{[\mathrm{IR}_8]}\}$, WMFC
170-173	Same as in Figure 7.21

7.25. Conditional branch

Address (Octal)	**Microinstruction**
000-002	Same as in Figure 7.21
003	μBranch $\{\mu\mathrm{PC} \leftarrow 300$
300	if Z+(N$\oplus$V = 1 then μBranch $\{\mu\mathrm{PC} \leftarrow 304\}$
301	PC_{out}, Y_{in}
302	$\mathrm{Address}_{out}$, SelectY, Add, Z_{in}
303	Z_{out}, PC_{in}, End

7.26. Assume microroutine starts at 300 for all three instructions. (Altenatively, the instruction decoder may branch to 302 directly in the case of an unconditional branch instruction.)

Address (Octal)	**Microinstruction**
000-002	Same as in Figure 7.21
003	μBranch $\{\mu\mathrm{PC} \leftarrow 300\}$
300	if Z+(N$\oplus$V = 1) then μBranch $\{\mu\mathrm{PC} \leftarrow 000\}$
301	if (N = 1) then μBranch $\{\mu\mathrm{PC} \leftarrow 000\}$
302	PC_{out}, Y_{in}
303	Offset-field-of-IR_{out}, SelectY, Add, Z_{in}
304	Z_{out}, PC_{in}, End

7.27. The answer to problem 3.26 holds in this case as well, with the restriction that one of the operand locations (either source or destination) must be a data register.

Address (Octal)	**Microinstruction**
000-002	Same as in Figure 7.21
003	μBranch $\{\mu\text{PC} \leftarrow 010\}$
010	if ($\text{IR}_{10-8} = 000$) then μBranch $\{\mu\text{PC} \leftarrow 101\}$
011	if ($\text{IR}_{10-8} = 001$) then μBranch $\{\mu\text{PC} \leftarrow 111\}$
012	if ($\text{IR}_{10-9} = 01$) then μBranch $\{\mu\text{PC} \leftarrow 121\}$
013	if ($\text{IR}_{10-9} = 10$) then μBranch $\{\mu\text{PC} \leftarrow 141\}$
014	μBranch $\{\mu\text{PC} \leftarrow 161\}$
121	Rsrc_{out}, MAR_{in}, Read, Select4, Add, $\text{Z}in$
122	Z_{out}, Rsrc_{in}
123	if ($\text{IR}_8 = 1$) then μBranch $\{\mu\text{PC} \leftarrow 171\}$
124	μBranch $\{\mu\text{PC} \leftarrow 170\}$
170-173	Same as in Figure 7.21

7.28. There is no change for the five address modes in Figure 7.20. Absolute and Immediate modes require a separate path. However, some sharing may be possible among absolute, immediate, and indexed, as all three modes read the word following the instruction. Also, Full Indexed mode needs to be implemented by adding the contents of the second register to generate the effective address. After each memory access, the program counter should be updated by 2, rather than 4, in the case of the 16-bit processor.

7.29. The same general structure can be used. Since the dst operand can be specified in any of the five addressing modes as the src operand, it is necessary to replicate the microinstructions that determine the effective address of an operand. At microinstruction 172, the source operand should placed in a temporary register and another tree of microinstructions should be entered to fetch the destination operand.

7.30. (*a*) A possible address assignment is as follows.

Address	**Microinstruction**
0000	A
0001	B
0010	if (b_6b_5) = 00) then μBranch 0111
0011	if (b_6b_5) = 01) then μBranch 1010
0100	if (b_6b_5) = 10) then μBranch 1100
0101	I
0110	μBranch 1111
0111	C
1000	D
1001	μBranch 1111
1010	E
1011	μBranch 1111
1100	F
1101	G
1110	H
1111	J

(*b*) Assume that bits b_{6-5} of IR are ORed into bit μPC_{3-2}

Address	**Microinstruction**
0000	A
0001	B; $\mu PC_{3-2} \leftarrow b_{6-5}$
0010	C
0011	D
0100	μBranch 1111
0101	E
0110	μBranch 1111
0111	F
1011	G
1100	H
1101	μBranch 1111
1110	I
1111	J

(*c*)

Address	**Microinstruction**	
	Next address	Function
0000	0001	A
0001	0010	B; $\mu\text{PC}_{3-2} \leftarrow b_{6-5}$
0010	0011	C
0011	1111	D
0110	1111	E
1010	1011	F
1011	1100	G
1100	1111	H
1110	1111	I
1111	–	J

7.31. Put the Y_{in} control signal as the fourth signal in F5, to reduce F3 by one bit. Combine fields F6, F7, and F8 into a single 2-bit field that represents:

00: Select4
01: SelectY
10: WMFC
11: End

Combining signals means that they cannot be issued in the same microinstruction.

7.32. To reduce the number of bits, we should use larger fields that specify more signals. This, inevitably, leads to fewer choices in what signals can be activated at the same time. The choice as to which signals can be combined should take into account what signals are likely to be needed in a given step.

One way to provide flexibility is to define control signals that perform multiple functions. For example, whenever MAR is loaded, it is likely that a read command should be issued. We can use two signals: MAR_{in} and $\text{MAR}_{in}\cdot$ Read. We activate the second one when a read command is to be issued. Similarly, Z_{in} is always accompanied by either Select Y or Select4. Hence, instead of these three signals, we can use $\text{Z}_{in} \cdot \text{Select4}$ and $\text{Z}_{in} \cdot \text{SelectY}$.

A possible 12-bit encoding uses three 4-bit fields FA, FB, and FC, which combine signals from Figure 7.19 as follows:

FA: F1 plus, $\text{Z}_{out}\cdot$ End, $\text{Z}_{out}\cdot$ WMFC. (11 signals)

FB: F2, F3, Instead of Z_{in}, MAR_{in}, and MDR_{in} use $\text{Z}_{in}\cdot\text{Select4}$, $\text{Z}_{in}\cdot\text{SelectY}$, MAR_{in}, $\text{MAR}_{in} \cdot \text{Read}$, and $\text{MDR}_{in}\cdot$ Write. (13 signals)

FC: F4 (16 signals)

With these choices, step 5 in Figure 7.6 must be split into two steps, leading to an 8-step sequence. Figure 7.7 remains unchanged.

7.33. Figure 7.8 contains two buses, A and B, one connected to each of the two inputs of the ALU. Therefore, two fields are needed instead of F1; one field to provide gating of registers onto bus A, and another onto bus B.

7.34. Horizontal microinstructions are longer. Hence, they require a larger microprogram memory. A vertical organization requires more encoding and decoding of signals, hence longer delays, and leads to longer microprograms and slower operation. With the high-density of today's integrated circuits, the vertical organization is no longer justified.

7.35. The main advantage of hardwired control is fast operation. The disadvantages include: higher cost, inflexibility when changes or additions are to be made, and longer time required to design and implement such units.

Microprogrammed control is characterized by low cost and high flexibility. Lower speed of operation becomes a problem in high-performance computers.