Chapter 5
Basic Processing Unit

5.1 The clock period must accommodate all propagation delay plus the setup time for registers.

(*a*) Minimum clock period = 70 + 600 + 50 = 720 ps

(*b*) Minimum clock period = 70 + 200 + 50 = 320 ps

5.2 Registers are always enabled for data to flow from one stage to the next. The information available at the input of a register is loaded into the register at the end of the clock cycle. When a new instruction is loaded in the instruction register in step 1, the contents of the registers selected by bit fields IR_{31-27} and IR_{26-22} are loaded into registers RA and RB, respectively, at the end of step 2. Before that, the contents of these registers are determined by the previous instruction.

Step	RA	RB	RZ	RM	RY
1	*	*	*	*	*
2	*	*	*	*	*
3	85320	4200	*	*	*
4	85320	4200	86320	4200	*
5	85320	4200	86320	4200	75900

* These values are determined by the previous instruction.

5.3 Knowing where the register fields are in the instruction makes it possible for the processor hardware to read the source registers before decoding the instruction. This is why it is possible to read the source registers in step 2 of Figures 5.11, for example.

5.4 Register contents are read in step 2 and loaded into the inter-stage registers at the end of that clock period.

Step	RA	RB	RZ	RY	R6
3	1000	2500	*	*	7500
4	1000	2500	−1500	*	7500
5	1000	2500	−1500	−1500	7500
1	1000	2500	−1500	−1500	−1500

* These values are determined by the previous instruction.

5.5 The result of the instruction is loaded into R4 at the end of step 5.

Step	PC	R4	RA	RM	RZ	RY
1	37C00	1000	*	*	*	*
2	37C04	1000	*	*	*	8
3	37C04	1000	1000	*	*	*
4	37C04	1000	1000	B2500	0	*
5	37C04	1000	1000	B2500	0	0
1	37C04	0	1000	B2500	0	0

* These values are determined by the previous instruction.

5.6 If $x_3 = 0$ and $y_3 = 1$, then X is greater than Y.
If $x_3 = 0$ and $y_3 = 0$, compare the remaining bits as for unsigned numbers.
If $x_3 = 1$ and $y_3 = 1$, compare the remaining bits as for unsigned numbers, but with individual bit values complemented.
Thus, the logic expressions for the three outputs may be written as follows:

$$\begin{aligned} XGY = \;& \overline{x}_3 y_3 + \overline{x}_3 \overline{y}_3 (x_2 \overline{y}_2 + \overline{(x_2 \oplus y_2)} \cdot (x_1 \overline{y}_1 + \overline{(x_1 \oplus y_1)}\, x_0 \overline{y}_0) \\ & + x_3 y_3 (\overline{x}_2 y_2 + \overline{(x_2 \oplus y_2)} \cdot (\overline{x}_1 y_1 + \overline{(x_1 \oplus y_1)}\, \overline{x}_0 y_0)) \\ XEY = \;& \overline{(x_2 \oplus y_2) \cdot (x_1 \oplus y_1) \cdot (x_0 \oplus y_0)} \\ XLY = \;& \overline{XGY + XEY} \end{aligned}$$

5.7 Register LINK needs to be read in step 2 of a return-from-subroutine instruction, before the instruction is decoded. The address LINK is placed in bits IR_{31-27} because the processor always reads the registers pointed to by these bits during step 2, without waiting for instruction decoding to be completed.

5.8 A return-from-interrupt instruction restores the contents of the PC in the same way as the return-from subroutine instruction in Example 5.4, but using register IRA instead of register LINK. It must also restore the contents of the PS from register IPS. The execution steps are:

1. Memory address ← [PC], Read memory, Wait for MFC, IR ← Memory data,
   PC ← [PC] + 4
2. Decode instruction, RA ← [IRA]
3. PC ← [RA], PS ← [IPS]
4. No action
5. No action

5.9 If register addresses are not in the same bit positions for all instructions, the processor needs to decode the instruction, at least partially, before being able to read its source registers. The source registers would then have to be read in step 3, adding a step to the execution sequence.

In order to maintain a five-cycle execution sequence, the clock cycle may be extended to enable the processor to read the source registers at the end of step 2, after the instruction has been sufficiently decoded to determine where the register addresses are in the IR. With this adjustment, a 5-step execution sequence can be used.

An alternative is to delay reading the source registers until step 3, and to send their contents directly to the ALU. Registers RA and RB would not be needed in this case. The clock cycle must be sufficiently long to accommodate a register access plus an ALU operation. Note that register access may start sometime towards the end of cycle 2, as soon as instruction decoding identifies the location of the register addresses in the IR.

Which of these two alternatives is preferable depends on design details and the associated delays.

5.10 Assume that the manner in which the immediate operand is extended is determined by a control input called SE. The operand is sign extended when SE =1, otherwise it is padded with zeros. The output Imm_{31-0} for the Immediate block may be derived as follows.

$Imm_0 = IR_6$, $Imm_1 = IR_7, \ldots,$ $Imm_{15} = IR_{21}$
$Imm_{16} = SE \cdot IR_{21}$, $Imm_{17} = SE \cdot IR_{21}, \ldots,$ $Imm_{31} = SE \cdot IR_{21}$

5.11 Assume that branch instructions are executed in five steps, even though no action is required in steps 4 and 5.

(*a*) Execution rate = $1 \times 10^9/5$ = 200 million instructions per second

(*b*) On average, instruction fetch takes $0.9 + 0.1 \times 4 = 1.3$ cycles. All instructions, except Load and Store, take four more cycles to complete. Load and Store instructions take two additional cycles, on average.
Average completion time = $1.3 + (0.2 + 0.5) \times 4 + (0.2 + 0.1) \times 6$ = 5.9 cycles
Instruction rate = $10^9/5.9$ = 169.5 million instructions per second

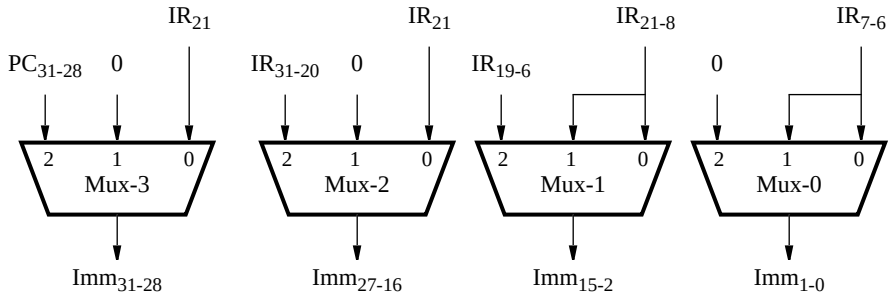
5.12 Computational instructions, which make up 50% of the instructions, are completed in four clock cycles.

Instruction rate $= 10^9/(0.5 \times 5 + 0.5 \times 4) = 222.2$ million instructions per second

5.13 The incremented program counter, [PC]+4, is available in step 2, and the location of the branch offset in the instruction is known. Hence, the updated value of the PC, [PC] + Branch offset, can be computed in step 2, just in case the instruction being decoded turns out to be a branch instruction. Of course, the updated value must not be loaded in the PC until the branch condition has been tested.

The branch condition can be tested in step 2 by feeding the two outputs of the register file directly to the comparator. The output of the comparator can then be used to load the branch target address into the PC at the end of step 2. In this case, three actions are being carried out sequentially: reading the register file, comparing the two operand, and loading the PC. The clock period must be sufficiently long to allow all three actions to be completed.

5.14 Different values for the output of the Immediate block, Imm_{31-0}, are selected by the Extend control input in Figure 5.19. Multiplexers may be used to select the values for different bit fields as shown in the figure below. Selections 0, 1, and 2 correspond to cases (*a*), (*b*), and (*c*), respectively.



5.15 The autoincrement addressing mode means that the contents of register R5 are incremented by 4 after being used to read the memory operand. Both the updated value of register R5 and the operand read from the memory need to be written into the register file. This is not possible in the hardware organization in Figure 5.8.

5.16 The Immediate block in Figure 5.9 implements the various ways in which immediate data in the IR are used by various instructions. For the OrHigh instruction, the Immediate block needs to shift the immediate data to the high-oder bit positions of its output word, Imm, as follows:

$$\text{Imm}_{31-16} = \text{IR}_{21-6}, \quad \text{Imm}_{15-0} = 0$$

After fetching the instruction: OrHigh R*i*, #Value, the execution steps are:

2. Decode instruction, RA ← R*i*
3. RZ ← [RA] OR Imm
4. RY ← [RZ]
5. R*i* ← [RY]

5.17 The PC is incremented in the clock cycle in which MFC is asserted.

Step	Cycle	MFC	PC
1	1	0	46000
1	2	0	46000
1	3	0	46000
1	4	1	46000
2	1	0	46004

5.18 For a MoveControl R*i*, IPS instruction, the execution steps are:

3. No action
4. RY ← [IPS]
5. R*i* ← [RY]

For a MoveControl IPS, R*i* instruction,

2. RA ← [R*i*]
3. IPS ← [RA]
4. No action
5. No action

5.19 Execution of this Subtract instruction requires the following sequence of actions to be performed:

- Read the value LOC from the second word of the instruction and hold it in a temporary register.
- Read location LOC in the memory.
- Perform the subtraction operation.
- Store the result in location LOC.

The hardware of Figure 5.8 implements a processor in which all instructions are fetched and executed in five steps, and each step is executed in one stage. The sequencing of the execution steps is defined by the hardware interconnections. The actions above do not follow that predefined sequence. This is not the case in Figure 5.22, which allows data to be transferred from any unit to any other unit at any time, as required by the instruction being executed. Also, the number of steps can be as large as needed.

5.20 Assume that the instruction decoder generates the following signals,one of which is set to 1 to identify the instruction being executed.

Rgr	All register-to-register instructions
Imd	Instructions that use immediate data
Mem	Load and Store instructions
Sbr	Subroutine call instructions

Assume that the two bits of C_select are C_select_0 and C_select_1. These control signals select the destination register of the instruction. They should be set to 01 for Rgr instructions, 10 for subroutine call instructions, and 00 otherwise. This leads to

$$C\_select_0 = Rgr, \quad C\_select_1 = Sbr$$

MA_select selects register Z as the source of the memory address, but only during step 4 (T4). It selects the PC at other times.

$$MA\_select = T4$$

Y_select should be equal to 1 for Load instructions, 2 for subroutine call instructions, and 0 otherwise.

$$Y\_select_0 = Mem, \quad Y\_select_1 = Sbr$$

Note that when a unit is not being used, the setting of its control signals is of no consequence. It is a "don't care" in the truth table of that signal. For example, MuxY is used only in step 5. The selection it makes at other times is immaterial. Hence, there is no need to include T5 in the logic expressions for Y_select.

5.21 The WMFC signal identifies a step in which either a Read-memory or a Write-memory command is issued. In these steps, advancing the counter must be delayed until the MFC signal is asserted.

5.22 The PC is enabled in step 1, when an instruction is being fetched from memory. If the instruction is not found in the cache and has to be read from the main memory, this step will involve several clock cycles. Without the MFC signal, the PC would be incorrectly incremented in every cycle.

5.23 MuxPC selects RA in step 3 (T3) of a Call_register instruction. It selects the adder output at all other times. MuxINC selects the immediate value in step 3, to be used in branch instructions.

$$\begin{aligned} \text{PC\_select} &= \overline{\text{T3} \cdot \text{Call\_Register}} \\ \text{INC\_select} &= \text{T3} \end{aligned}$$

5.24 (*a*) The starting address of a microroutine is generated as a result of decoding an instruction. The delays incurred by the steps in Figure 5.26 are given in the following table. Assume that incrementing the PC requires less time than reading a word from the cache.

Step	Delay components	Minimum time
1	1.5 + 1.7	3.2
2	1.5 + 2.1	3.6
3	1.5 + 1.7	3.2
4	1.5 + 2.2	3.7
5	1.5 + 1.7	3.2
6	1.5 + 2.2	3.7
7	1.5 + 1.7	3.2

(*b*) Minimum clock cycle = 3.7 ns

5.25 The actions needed after decoding the instruction are:

3. Memory address ← [R5], Read memory, Wait for MFC, Temp1 ← Memory data
4. R5 ← [R5] + 4
5. R3 ← [Temp1]

5.26 Assume that the instruction

Call-Register R5, R7

calls a subroutine whose address is given in register R5 and stores the return address on a stack pointed to by R7. After reading and decoding the instruction, the following actions are needed.

3. Temp1 ← [PC] + 0
4. PC ← [R5] + 0
5. R7 ← [R7] − 4
6. Memory address ← [R7], Memory data ← [Temp1], Write memory, Wait for MFC