# Exercise 1

1.

Table, calendar

Description automatically generated

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Instr |  |  |  |  |  |  |  |  |
| R-type | Write/read PC (2ns) | Instruction memory (15ns) | Control (4ns) | Mux (ALUScr) (2ns) | ALU (10ns) | Mux (MemtoReg) (2ns) | Register write (5ns) |  |
|  | Add (PC) (8ns) | Register read (7ns) | ALU Control (2ns) |  | Gate (1ns) + Mux branch (2ns) |  |  |
|  |  |  |  |  |  |  |  |
| Load | Write/read PC (2ns) | Instruction memory (15ns) | Control (4ns) | Mux (ALUScr) (2ns) | ALU (10ns) | Data memory access (15ns) | Mux (MemtoReg) (2ns) | Register write (5ns) |
|  | Add (PC) (8ns) | Register read (7ns) | ALU Control (2ns) |  | Gate (1ns) + Mux branch (2ns) | Mux(RegDst) (2ns) |  |
|  |  | Sign extend (3ns) |  |  |  |  |  |
| Store | Write/read PC (2ns) | Instruction memory (15ns) | Control (4ns) | Mux (ALUScr) (2ns) | ALU (10ns) | Data memory access (15ns) |  |  |
|  | Add (PC) (8ns) | Register read (7ns) | ALU Control (2ns) |  | Gate (1ns) + Mux branch (2ns) |  |  |
|  |  | Sign extend (3ns) |  |  |  |  |  |
| Branch | Write/read PC (2ns) | Instruction memory (15ns) | Control (4ns) | Mux (ALUScr) (2ns) | ALU (10ns) | Gate (1ns) + Mux branch (2ns) |  |  |
|  | Add (PC) (8ns) | Register read (7ns) | ALU Control (2ns) | Add (PC + branch offset) (8ns) |  |  |  |
|  |  | Sign extend (3ns) | Shift left 2 (3ns) |  |  |  |  |

2.

R-type critical path latency = 2+15+7+2+10+1+2+5 = 44 ns

Load critical path latency = 2+15+7+2+10+15+2+5 = 58 ns

Store critical path latency = 2+15+7+2+10+15 = 51 ns

Branch critical path latency = 2+15+7+3+10+1+2 = 40 ns

3. Clock frequency should account for the longest critical path

**So F\_clk = 1/T\_longest = 1/ (58 ns) = 17,24 MHz**

4. The latency of each of the five stages is:

IF: t = max(PC + ADD, PC + Instruction mem) = max( 2+8, 2+15) = 17 ns

ID: t = max(Register file read , Sign extend) = max(7, 3) = 7 ns

EX: t = max(mux (ALUSrc) + ALU, ALU control + ALU, shift left + Add, mux (RegDst)) = max(2 + 10, 2 + 10, 3 + 8, 2) = 12 ns

MEM: t = max(Data memory access, gate+mux (PCSrc)) = max(15, 1+2) = 15 ns  
WB: t = max(mux + write register) = max(2+5) = 7 ns

Longest stage is IF with 17 ns. But also taking 10% overhead into account gives:

So T\_max = 17 ns \* 1,1 = 18,7 ns

**So F\_clk = 1 / T\_max = 1 / (18,7 ns) = 53,48 MHz**

# Exercise 2

1.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Q1 |  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
|  | lw R1, 0(R3) | F | D | E | M | W |  |  |  |  |  |  |  |  |
|  | lw R2, 0(R4) |  | F | D | E | M | W |  |  |  |  |  |  |  |
|  | beq R1, R2, target |  |  | F | D | E | M | W |  |  |  |  |  |  |
|  | add R3, R3, R4 |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | addi R4, R3, 4 |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | exit |  |  |  |  |  |  |  |  |  |  |  |  |  |
| target: | lw R5, 0(R4) |  |  |  |  |  | F | D | E | M | W |  |  |  |
|  | add R5, R5, R6 |  |  |  |  |  |  | F | D | E | M | W |  |  |

2.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Q2 |  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
|  | lw R1, 0(R3) | F | D | E | M | W |  |  |  |  |  |  |  |  |
|  | lw R2, 0(R4) |  | F | D | E | M | W |  |  |  |  |  |  |  |
|  | beq R1, R2, target |  |  | F | D (A2') | \*\* | E | M | W |  |  |  |  |  |
|  | add R3, R3, R4 |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | addi R4, R3, 4 |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | exit |  |  |  |  |  |  |  |  |  |  |  |  |  |
| target: | lw R5, 0(R4) |  |  |  |  |  |  | F | D | E | M | W |  |  |
|  | add R5, R5, R6 |  |  |  |  |  |  |  | F | D (A2") | \*\* | E | M |  |

3.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Q3 |  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
|  | lw R1, 0(R3) | F | D | E | M | W |  |  |  |  |  |  |  |  |
|  | lw R2, 0(R4) |  | F | D | E | M | W |  |  |  |  |  |  |  |
|  | beq R1, R2, target |  |  | F | D (A2''') | \*\* | E | M | W |  |  |  |  |  |
|  | add R3, R3, R4 |  |  |  | F (A3''') | -- | -- | -- | -- |  |  |  |  |  |
|  | addi R4, R3, 4 |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | exit |  |  |  |  |  |  |  |  |  |  |  |  |  |
| target: | lw R5, 0(R4) |  |  |  |  | F | D | E | M | W |  |  |  |  |
|  | add R5, R5, R6 |  |  |  |  |  | F | D (A2''') | \*\* | E | M |  |  |  |

4.

Possible branch delay slot usage:

From before: Not possible because of data dependencies from load words

From target: Move “lw R5, 0(R4)” up. So it becomes “beq” followed by “lw” followed by “add R3, R3, R4”

From not-taken fall-through: You can keep it as it is, the “add R3, R3, R4” is possible for the branch-delay slot