مستندسازی کامل و صریح ریزعملیات‌ها (RTL)

در این مستند، چرخه اجرای هر یک از دستورالعمل‌های الزامی پروژه به تفکیک ۵ فاز اصلی و به صورت کاملاً مستقل شرح داده شده است.

فازهای کلی اجرا

1. **واکشی (Fetch):** خواندن دستور از حافظه و افزایش شمارنده برنامه.
2. **رمزگشایی (Decode):** خواندن مقادیر از فایل ثبات‌ها و استخراج مقادیر بلافصل.
3. **اجرا (Execute):** انجام محاسبات توسط واحد محاسبه و منطق (ALU).
4. **حافظه (Memory):** دسترسی به حافظه داده (خواندن یا نوشتن).
5. **نوشتن (Write-back):** نوشتن نتیجه نهایی در فایل ثبات‌ها.

۱. دستورالعمل‌های محاسباتی/منطقی (R-type)

add rd, rs1, rs2

- **Fetch:** IR <- Mem[PC], PC <- PC + 4
- **Decode:** A <- Regs[rs1], B <- Regs[rs2]
- **Execute:** ALUResult <- A + B
- **Memory:** *(بدون عملیات)*
- **Write-back:** Regs[rd] <- ALUResult

sub rd, rs1, rs2

- **Fetch:** IR <- Mem[PC], PC <- PC + 4
- **Decode:** A <- Regs[rs1], B <- Regs[rs2]
- **Execute:** ALUResult <- A - B
- **Memory:** *(بدون عملیات)*
- **Write-back:** Regs[rd] <- ALUResult

and rd, rs1, rs2

- **Fetch:** IR <- Mem[PC], PC <- PC + 4
- **Decode:** A <- Regs[rs1], B <- Regs[rs2]
- **Execute:** ALUResult <- A & B
- **Memory:** *(بدون عملیات)*
- **Write-back:** Regs[rd] <- ALUResult

or rd, rs1, rs2

- **Fetch:** IR <- Mem[PC], PC <- PC + 4
- **Decode:** A <- Regs[rs1], B <- Regs[rs2]
- **Execute:** ALUResult <- A | B
- **Memory:** *(بدون عملیات)*

- **Write-back:** Regs[rd] <- ALUResult

xor rd, rs1, rs2

- **Fetch:** IR <- Mem[PC], PC <- PC + 4
- **Decode:** A <- Regs[rs1], B <- Regs[rs2]
- **Execute:** ALUResult <- A ^ B
- **Memory:** *(بدون عملیات)*
- **Write-back:** Regs[rd] <- ALUResult

slt rd, rs1, rs2

- **Fetch:** IR <- Mem[PC], PC <- PC + 4
- **Decode:** A <- Regs[rs1], B <- Regs[rs2]
- **Execute:** ALUResult <- (A < B) ? 1 : 0 (مقایسه علامت‌دار)
- **Memory:** *(بدون عملیات)*
- **Write-back:** Regs[rd] <- ALUResult

۲. دستورالعمل‌های محاسباتی با مقدار بلافصل (I-type)

addi rd, rs1, imm

- **Fetch:** IR <- Mem[PC], PC <- PC + 4
- **Decode:** A <- Regs[rs1], imm <- sign-extend(IR[31:20])
- **Execute:** ALUResult <- A + imm
- **Memory:** *(بدون عملیات)*
- **Write-back:** Regs[rd] <- ALUResult

۳. دستورالعمل‌های دسترسی به حافظه (Load/Store)

lw rd, offset(rs1) (Load Word)

- **Fetch:** IR <- Mem[PC], PC <- PC + 4
- **Decode:** A <- Regs[rs1], imm <- sign-extend(offset)
- **Execute:** ALUResult <- A + imm (محاسبه آدرس)
- **Memory:** ReadData <- Mem[ALUResult]
- **Write-back:** Regs[rd] <- ReadData

sw rs2, offset(rs1) (Store Word)

- **Fetch:** IR <- Mem[PC], PC <- PC + 4
- **Decode:** A <- Regs[rs1], B <- Regs[rs2], imm <- sign-extend(offset)
- **Execute:** ALUResult <- A + imm (محاسبه آدرس)
- **Memory:** Mem[ALUResult] <- B
- **Write-back:** *(بدون عملیات)*

۴. دستورالعمل‌های انشعاب شرطی (B-type)

beq rs1, rs2, label (Branch if Equal)

- **Fetch:** IR <- Mem[PC], PC_temp <- PC + 4
- **Decode:** A <- Regs[rs1], B <- Regs[rs2], imm <- sign-extend(label_offset)
- **Execute:** ALUResult <- A - B, ZeroFlag <- (ALUResult == 0)
- **Memory / PC Update:** if (ZeroFlag) then PC <- PC_temp + imm else PC <- PC_temp
- **Write-back:** *(بدون عملیات)*

bne rs1, rs2, label (Branch if Not Equal)

- **Fetch:** IR <- Mem[PC], PC_temp <- PC + 4
- **Decode:** A <- Regs[rs1], B <- Regs[rs2], imm <- sign-extend(label_offset)
- **Execute:** ALUResult <- A - B, ZeroFlag <- (ALUResult == 0)
- **Memory / PC Update:** if (!ZeroFlag) then PC <- PC_temp + imm else PC <- PC_temp
- **Write-back:** *(بدون عملیات)*

۵. دستورالعمل‌های پرش (Jump)

jal rd, label (Jump and Link)

- **Fetch:** IR <- Mem[PC], PC_temp <- PC + 4
- **Decode:** imm <- sign-extend(label_offset)
- **Execute:** PC <- PC + imm (محاسبه آدرس پرش)
- **Memory:** *(بدون عملیات)*
- **Write-back:** Regs[rd] <- PC_temp (ذخیره آدرس بازگشت)

jalr rd, offset(rs1)

- **Fetch:** IR <- Mem[PC], PC_temp <- PC + 4
- **Decode:** A <- Regs[rs1], imm <- sign-extend(offset)
- **Execute:** PC <- (A + imm) & 0xFFFFFFFE (محاسبه آدرس پرش و صفر کردن بیت آخر)
- **Memory:** *(بدون عملیات)*
- **Write-back:** Regs[rd] <- PC_temp

۶. دستورالعمل‌های بارگذاری مقدار بالا (U-type)

lui rd, imm (Load Upper Immediate)

- **Fetch:** IR <- Mem[PC], PC <- PC + 4
- **Decode:** imm <- IR[31:12]
- **Execute:** ALUResult <- imm << 12
- **Memory:** *(بدون عملیات)*
- **Write-back:** Regs[rd] <- ALUResult

auipc rd, imm (Add Upper Immediate to PC)

- **Fetch:** IR <- Mem[PC], PC_temp <- PC , PC <- PC + 4
- **Decode:** imm <- IR[31:12]
- **Execute:** ALUResult <- PC_temp + (imm << 12)
- **Memory:** *(بدون عملیات)*
- **Write-back:** Regs[rd] <- ALUResult

۷. دستورالعمل‌های ضرب و تقسیم (RV32M)

mul rd, rs1, rs2

- **Fetch:** IR <- Mem[PC], PC <- PC + 4
- **Decode:** A <- Regs[rs1], B <- Regs[rs2]
- **Execute:** ALUResult <- (A * B)[31:0] (بیت پایین حاصلضرب ۳۲)
- **Memory:** *(بدون عملیات)*
- **Write-back:** Regs[rd] <- ALUResult

div rd, rs1, rs2

- **Fetch:** IR <- Mem[PC], PC <- PC + 4
- **Decode:** A <- Regs[rs1], B <- Regs[rs2]
- **Execute:** ALUResult <- A / B (تقسیم صحیح علامت‌دار)
- **Memory:** *(بدون عملیات)*
- **Write-back:** Regs[rd] <- ALUResult

rem rd, rs1, rs2

- **Fetch:** IR <- Mem[PC], PC <- PC + 4
- **Decode:** A <- Regs[rs1], B <- Regs[rs2]
- **Execute:** ALUResult <- A % B (باقیمانده تقسیم علامت‌دار)
- **Memory:** *(بدون عملیات)*
- **Write-back:** Regs[rd] <- ALUResult