Chapter1

1. Consider two different implementations of the same instruction set architecture. The instructions can be divided into four classes according to their CPI (classes A, B, C, and D). P1 with a clock rate of 2.5GHz and CPIs of 1, 2, 3, and 3, and P2 with a clock rate of 3GHz and CPIs of 2, 2, 2, and 2. Given a program with a dynamic instruction count of 1.0E6 instructions divided into classes as follows: 10% class A, 20% class B, 50% class C, and 20% class D, which is faster: _P2_ (P1 or P2)?
   1) What is the global CPI for each implementation?
      CPI(P1)=_ 2.6 _ CPI(P2)=_ 2.0 _
   2) Find the clock cycles required in both cases.
      Clock cycles(P1)=_26_*10^5 Clock cycles(P2)=_20_*10^5

2. Compilers can have a profound impact on the performance of an application. Assume that for a program, compiler A results in a dynamic instruction count of 1.0E9 and has an execution time of 1.1 s, while compiler B results in a dynamic instruction count of 1.2E9 and an execution time of 1.5 s.
   1) Find the average CPI for each program given that the processor has a clock cycle time of 1ns.
      CPI(A)=_ 1.1 _ CPI(B)=_ 1.25 _
   2) Assume the compiled programs run on two different processors. If the execution times on the two processors are the same, how much faster is the clock of the processor running compiler A’s code versus the clock of the processor running compiler B’s code?
      f_B/f_A=_ 1.37 _
   3) A new compiler is developed that uses only 6.0E8 instructions and has an average CPI of 1.1. What is the speedup of using this new compiler versus using compiler A or B on the original processor?
      T_A/T_{new}=_1.67_ T_B/T_{new}=_2.27_

Chapter2

1. Assemble:To convert the RISCV instructions into machine code.

Address(Hex)	RISCV Assembly Instruction	Machine Code(Hex)

200000	jal x0, L1	**0x070000006F**
200004	Loop: add t3, s3, s4	**0x01498E33**
200008	bne t3, t4, Loop	**0xFFDE1EE3**
……		-
200070	L1： ……	-

2. To convert the pseudoinstruction(left) into the shortest sequence of RISCV instructions.

Pseudoinstruction	**Function**	**RISCV instructions**
J Lable	goto Lable	**jal x0,Lable**
Seqz rd,rs	Rd = (rs==0)?1:0	**sltiu rd,rs,1**
Not rd,rs	rd = ~rs	**xori rd,rs,-1**

Chapter3

1. The data have no inherent meaning, different data format or different representation **have different meaning or value, please fill each blank.**

(A4 C0 00 6F) $_{16}$ is a two's complement integer	Its value in hexadecimal notation and signed and magnitude format is hexadecimal notation: -5B3FFF91 signed and magnitude format:1101 1011 0011 1111 1111 1111 1001 0001 .
(A4 C0 00 6F) $_{16}$ is a one's complement integer	Its value in hexadecimal notation and signed and magnitude format is: hexadecimal notation: -5B3FFF91 . signed and magnitude format:1101 1011 0011 1111 1111 1111 1001 0000
(04 C0 00 6F) $_{16}$ is a IEEE754 single precision floating-point number.	Expressed as decimal exponent and 7-digit hexadecimal mantissa, then its value is _ $1.40006F \times 2^{-118}$.
(04 C0 00 6F) $_{16}$ is a RISCV instruction.	This instruction is jal,x0,76 .

2. **A and B are the floating-point number with IEEE754 single precision. Write down the Binary representation of c.**

   A=C20E6666

   B=25.1   C=A+B

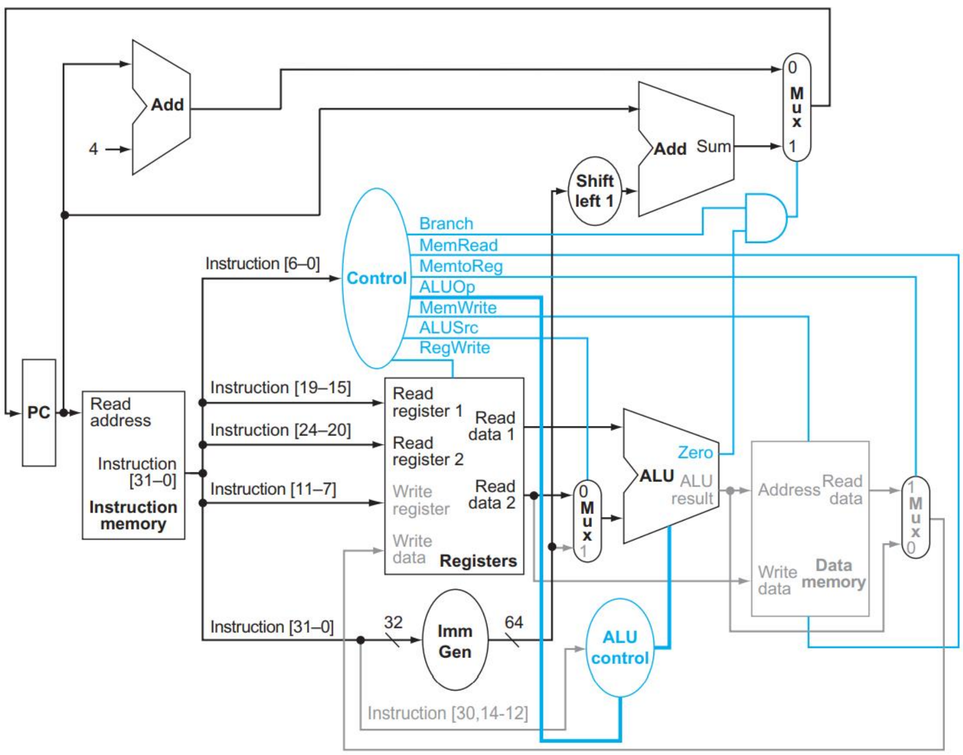
   1 10000010 01001111111111111111110

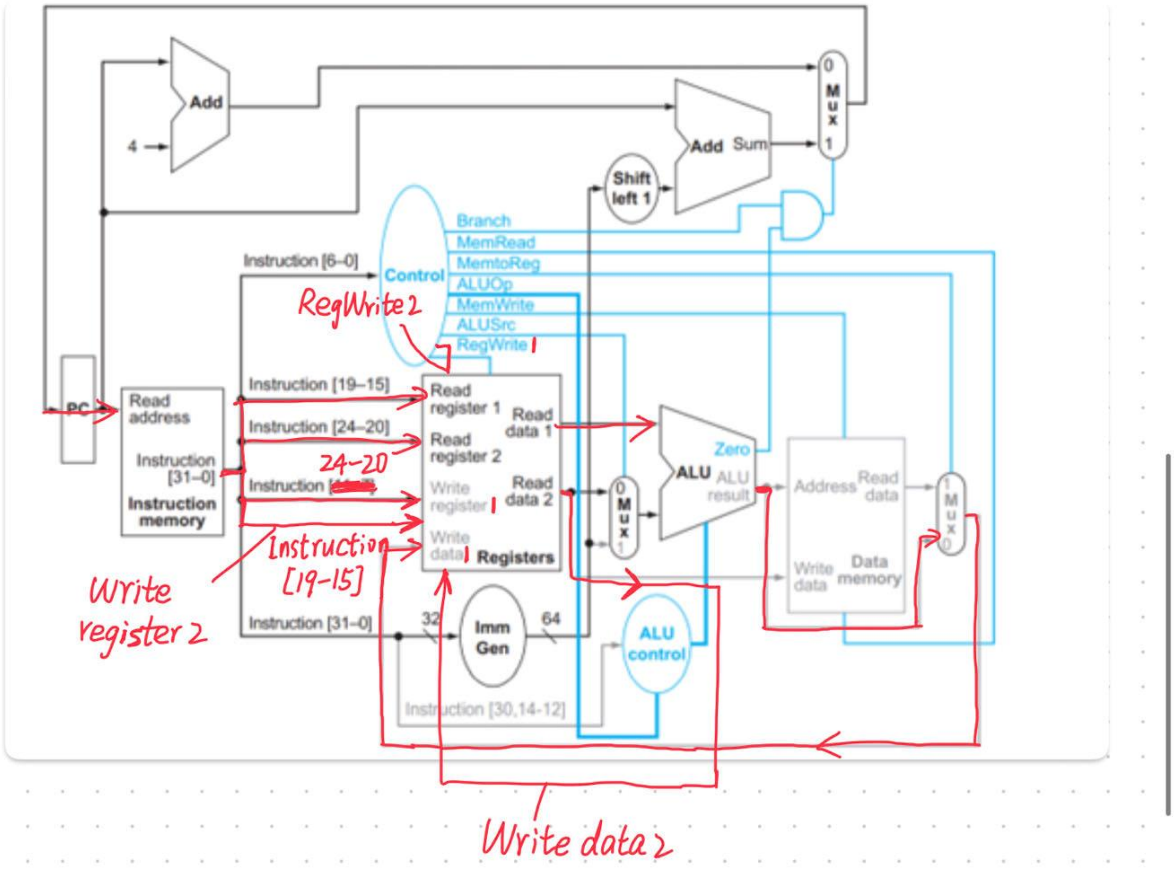
Chapter4

1. **Examine the difficulty of adding a proposed swap rs1, rs2 instruction to RISC-V. Interpretation:**

Reg[rs2]=Reg[rs1]; Reg[rs1]=Reg[rs2]

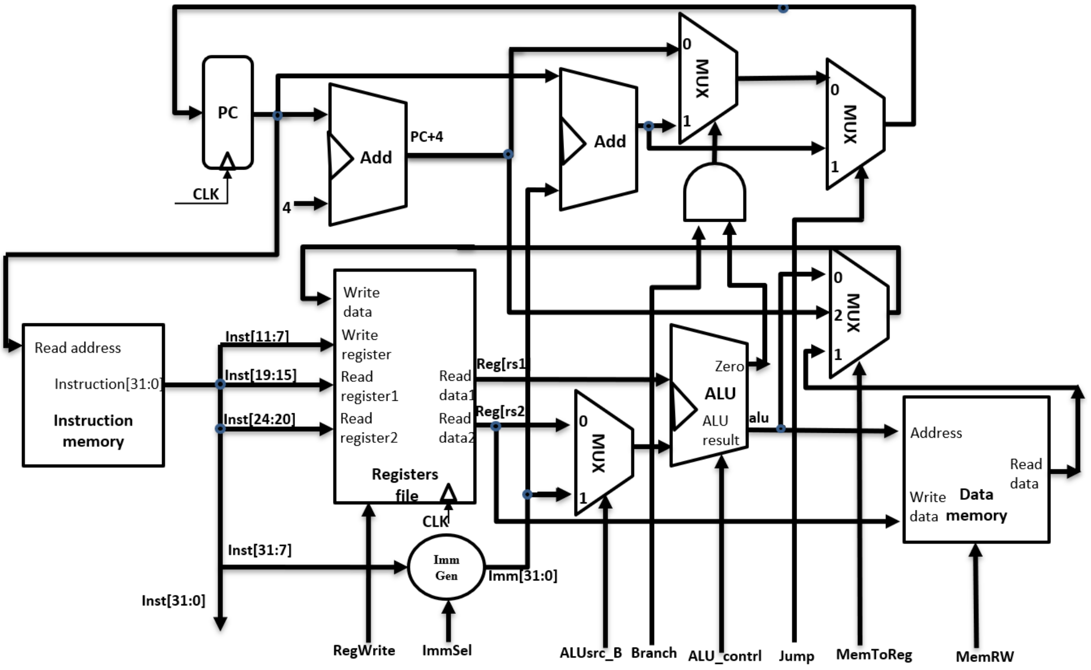
1) Which new functional blocks (if any) do we need for this instruction?
   No new functional blocks are needed.
2) Which existing functional blocks (if any) require modification?
   The register file needs to be modified so that it can write to two registers in the same cycle.
   The ALU also needs to be modified to allow read data 1 or 2 to be passed through to write data 1.
3) What new data paths do we need (if any) to support this instruction?
   A new data path to write data 2 is needed.
4) What new signals do we need (if any) from the control unit to support this instruction?
   A second RegWrite control wire is needed.
5) Modify Figure to demonstrate an implementation of this new instruction


Add
4
Add
Sum
Mux
0
1
Shift left 1
Instruction [6–0]
Control
Branch
MemRead
MemtoReg
ALUOp
MemWrite
ALUSrc
RegWrite
PC
Read address
Instruction [31–0]
Instruction memory
Instruction [19–15]
Instruction [24–20]
Instruction [11–7]
Read register 1
Read register 2
Write register
Write data
Read data 1
Read data 2
Registers
Mux
ALU
Zero
ALU result
Address
Read data
Write data
Data memory
Instruction [31–0]
32
Imm Gen
64
ALU control
Instruction [30,14-12]


Add
4
Add
Sum
Mux
Shift left 1
Instruction [6–0]
Control
Branch
MemRead
MemtoReg
ALUOp
MemWrite
ALUSrc
RegWrite 1
RegWrite 2
PC
Read address
Instruction [31–0]
Instruction memory
Instruction [19–15]
Instruction [24–20]
24-20
Instruction ~~[11–7]~~
Read register 1
Read register 2
Write register 1
Write data 1
Read data 1
Read data 2
Registers
Instruction [19-15]
Write register 2
ALU
Zero
ALU result
Address
Read data
Write data
Data memory
Instruction [31–0]
32
Imm Gen
64
ALU control
Instruction [30,14-12]
Write data 2

2. A new instruction is added to 7-instruction single-cycle CPU of RISC-V:
ss rs2, rs1, imm (Store Sum)
Interpretation: **Mem[Reg[rs2]]=Reg[rs1]+immediate**

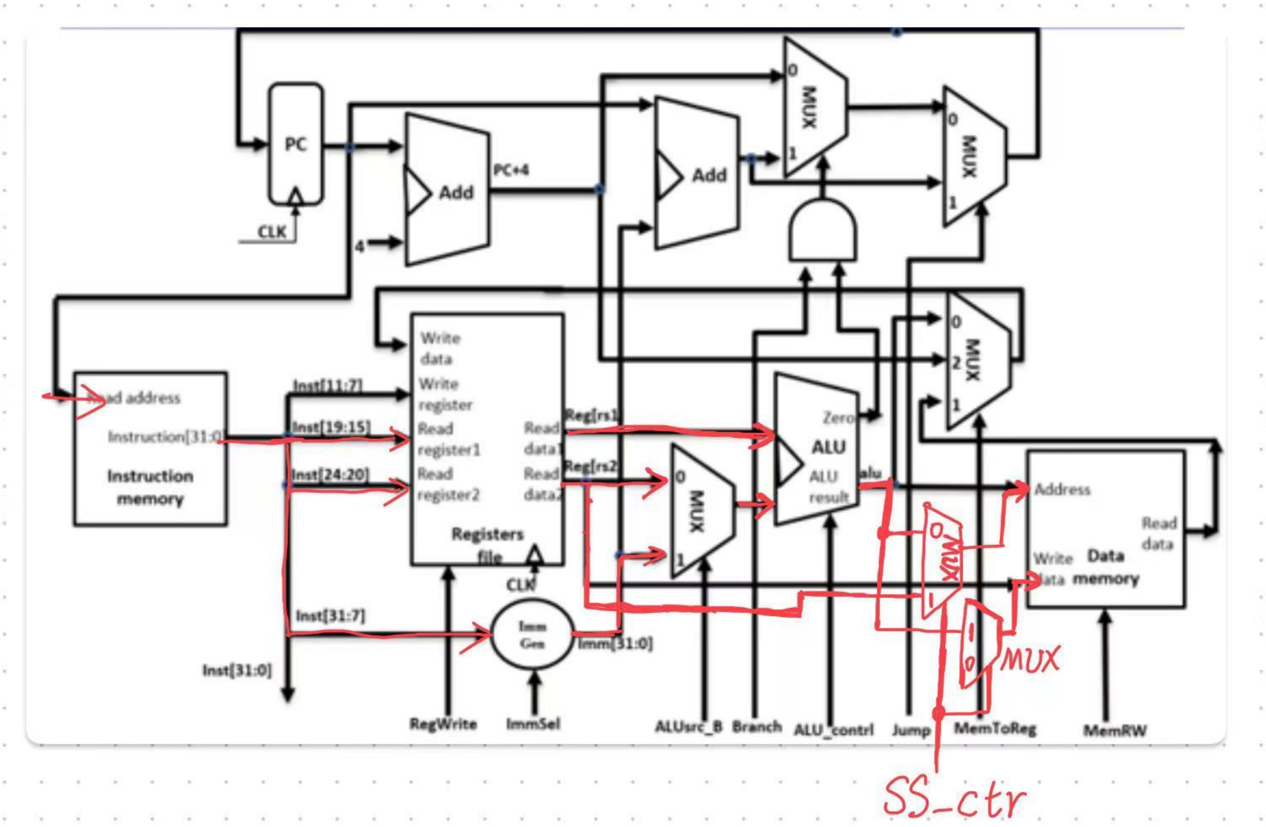
Picture 1



1) Which new functional blocks(if any) do we need for this instruction? Here functional blocks appear in above figure, they are: Instruction Memory, Data memory, PC, Adder, Register file, MUX,And-Gate.
We need some additional MUXes.

2) Which existing functional blocks(if any) require modification?
No functional blocks need to be modified.

3) Design datapath: Modify the picture1 to demonstrate an implementation of this new instruction. Suppose a new control signal SS-ctr is added, SS-ctr is 1 only when SS(Store Sum) is executing.



4) Design control signal: to set new signals to support this instruction;

Instruction	ALUSrcB	Memto-Reg	Reg Write	Mem Read	Mem Write	Branch	Jump	ALU_{Op1}	ALU_{Op0}	SS-ctr
SS	1	0	0	0	1	0	0	1	0	1

3. Add NOP instructions to the code below so that it will run correctly on a pipeline that does not handle data hazards.

```
addi x11, x12, 5
add x13, x11, x12
addi x14, x11, 15
add x15, x13, x12
```

```
addi x11, x12, 5
NOP
NOP
add x13, x11, x12
addi x14, x11, 15
NOP
add x15, x13, x12
```

4. Consider the fragment of RISC-V assembly below:

```
sd x29, 12(x16)
ld x29, 8(x16)
sub x17, x15, x14
beqz x17, label
add x15, x11, x14
sub x15, x30, x14
```

Suppose we modify the pipeline so that it has only one memory (that handles both instructions and data). In this case, there will be a structural hazard every time a program needs to fetch an instruction during the same cycle in which another instruction accesses data.

1) Draw a pipeline diagram to show were the code above will stall.

```
sd    x29, 12(x16)    IF ID EX ME WB
ld   x29, 8(x16)         IF ID EX ME WB
sub x17, x15, x14          IF ID EX ME WB
                             stall-----------
                                stall-----------
bez x17, label                       IF ID EX ME WB
add x15, x11, x14                       IF ID EX ME WB
sub x15,x30,x14                               IF ID EX ME WB
```

2) In general, is it possible to reduce the number of stalls/NOPs resulting from this structural hazard by reordering code?

   No.

3) Must this structural hazard be handled in hardware? We have seen that data hazards can be eliminated by adding NOPs to the code. Can you do the same with this structural hazard? If so, explain how. If not, explain why not.

   Yes.

   No.

   That's because NOPs also need to be fetched from instruction memory.