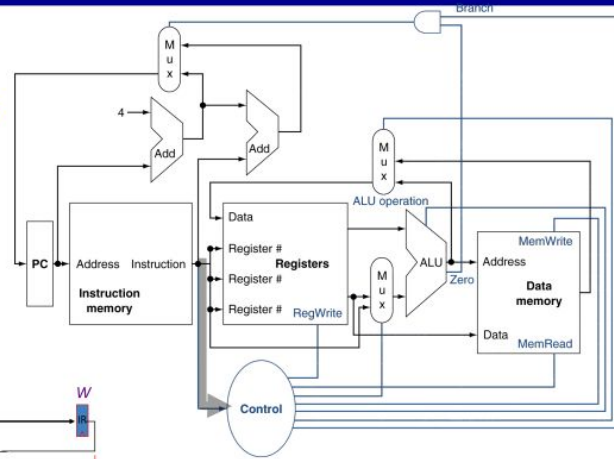
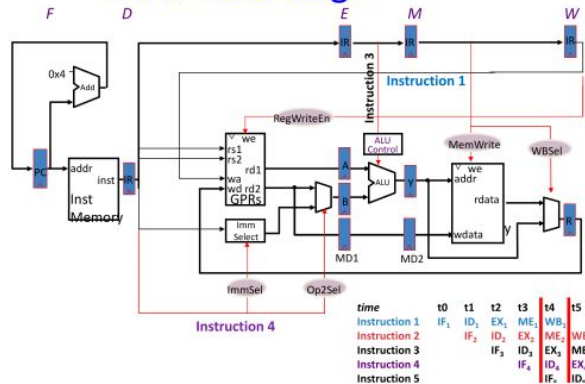


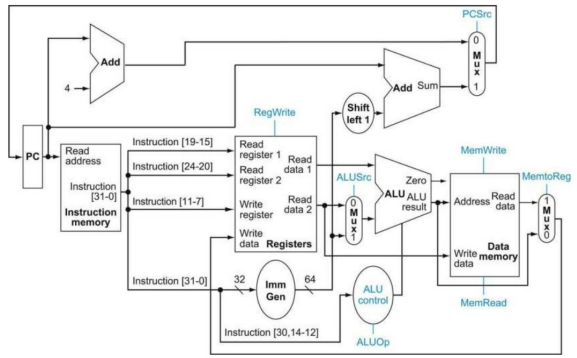
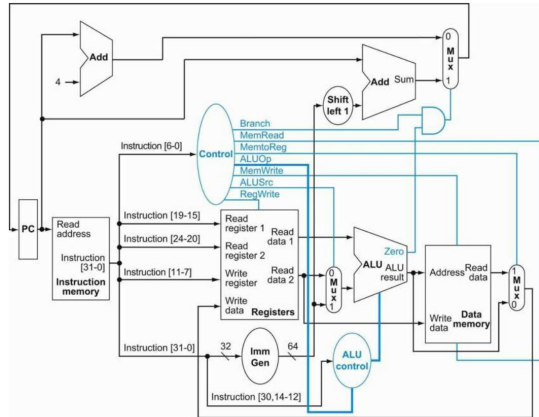
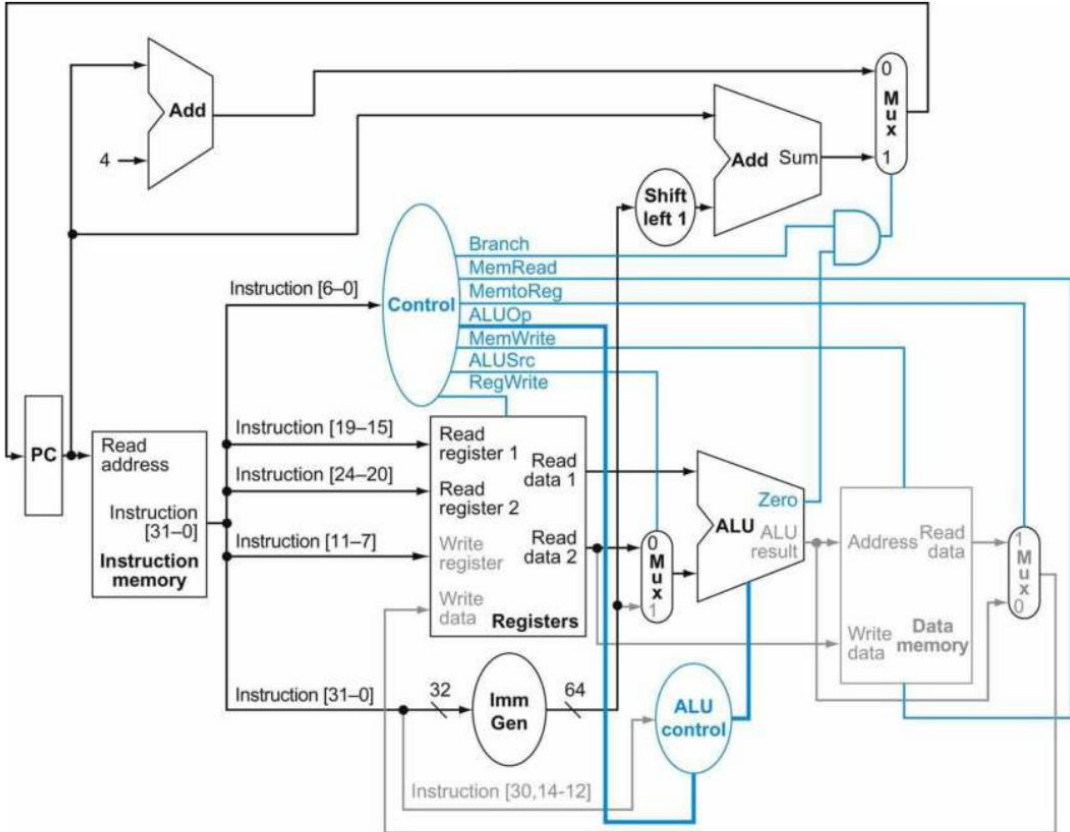
1. Wrong PC in branch circuit (slides)

Compared With Control Logic in Unpipelined

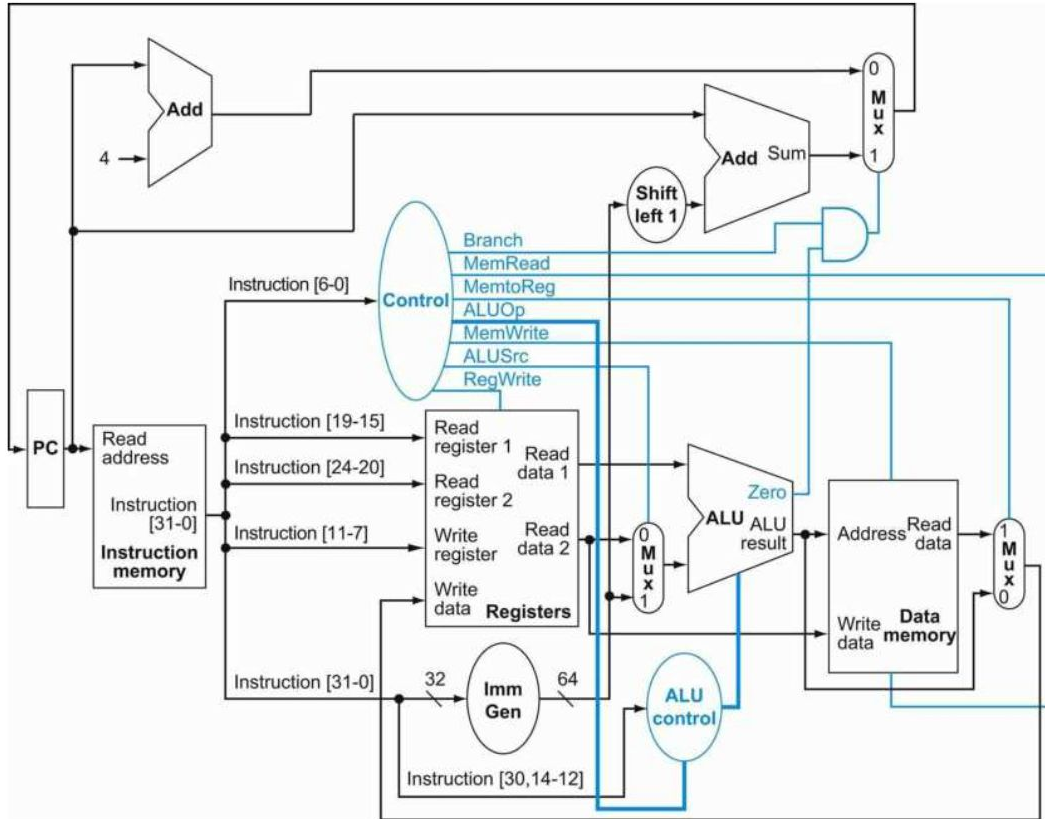
- Unpipelined:
 - Single control logic uses the instruction from IF
- Pipelined:
 - Distributed logics that uses instructions from IRs in each stage



1. Correct in book



2. Doubt in bit shifting PC value



Ma'am, in chapter 4, it is mentioned that the offset field is shifted left by 1 bit, making it a half-word offset. This shift supposedly doubles the effective range of the offset field. Could you please explain how this works?

Sir/maam kindly clarify on the fact that why in risc 5 we have a shifted address in the branch instruction by 1 and then again in the hardware we shift left 1 bit to make it aligned to the PC why not shift left 2 bits in the hardware itself as done in MIPS

We have 12 bits in immediate type instructions. Allowing us to move from $-(2^{11})$ to $+(2^{11})$ relative to the current address. But in the book "What is the range of byte addresses for conditional branches in RISC-V" the answer is -4096 to +4096. Please help me with this.

2. Why is there a left shift at all? (MIPS 2 bits, RISC5 1 bit)

2. RISC V compressed extension – 16 bit instructions

<https://www2.eecs.berkeley.edu/Pubs/TechRpts/2015/EECS-2015-209.pdf>

1.1 Introduction

This excerpt from the RISC-V User-Level ISA Specification describes the current draft proposal for the RISC-V standard compressed instruction set extension, named “C”, which reduces static and dynamic code size by adding short 16-bit instruction encodings for common operations. The C extension can be added to any of the base ISAs (RV32, RV64, RV128), and we use the generic term “RVC” to cover any of these. Typically, 50%–60% of the RISC-V instructions in a program can be replaced with RVC instructions, resulting in a 25%–30% code-size reduction.

We believe this draft represents the close to final design for RV32C and RV64C (it seems premature to freeze R128C), though we are requesting one more round of comments, hence the 1.9 revision number. Please send your comments to the `isa-dev` mailing list at `isa-dev@lists.riscv.org`.

1.2 Overview

RVC uses a simple compression scheme that offers shorter 16-bit versions of common 32-bit RISC-V instructions when:

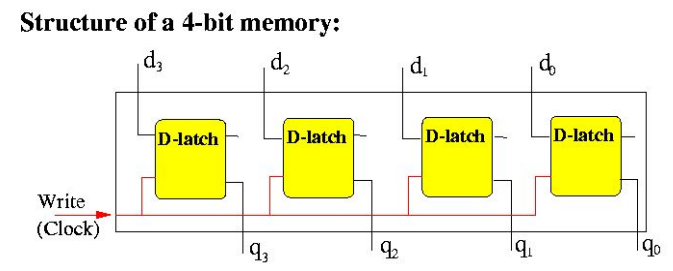
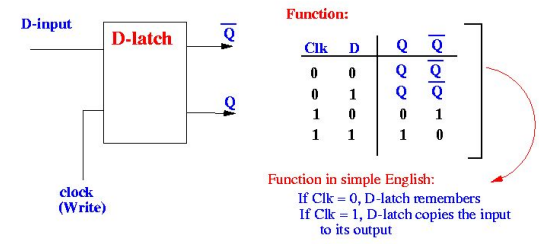
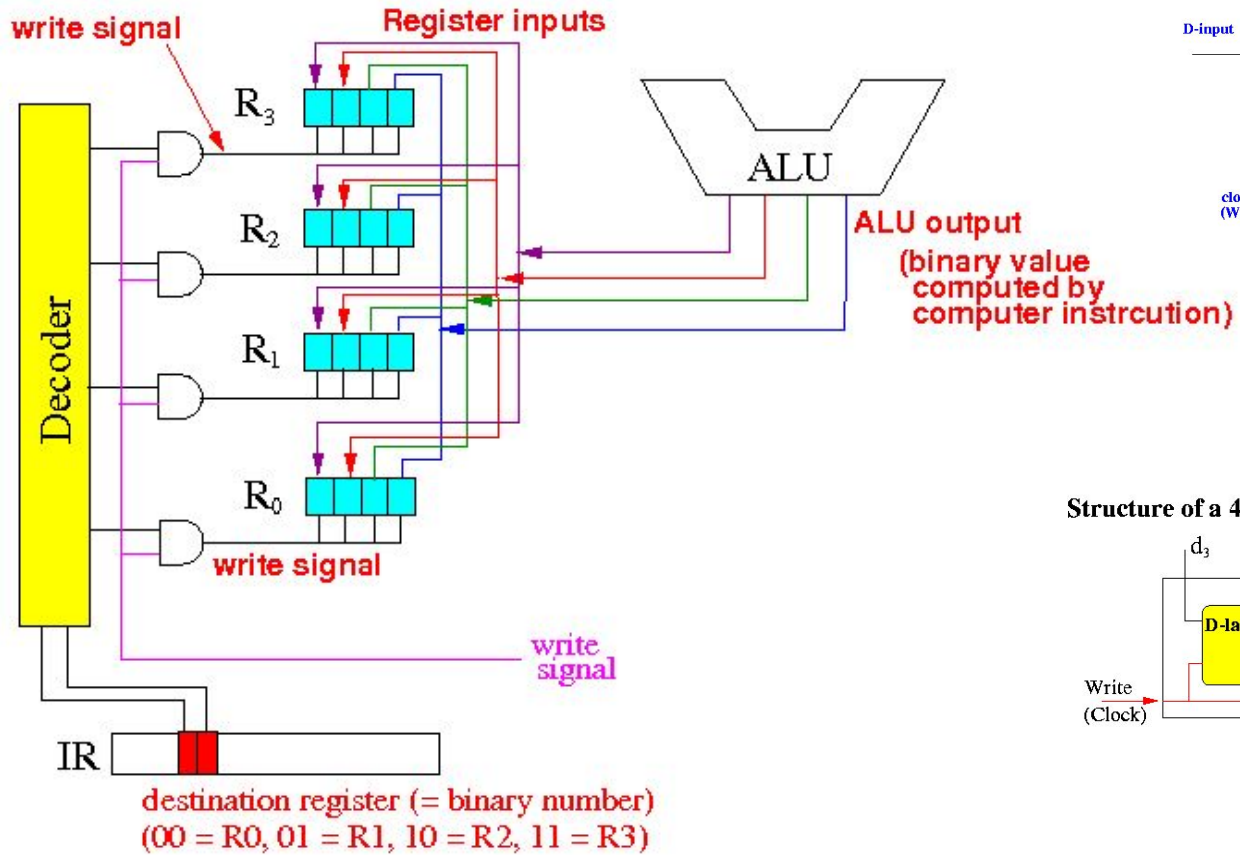
- the immediate or address offset is small, or
- one of the registers is the zero register (`x0`), the ABI link register (`x1`), or the ABI stack pointer (`x2`), or
- the destination register and the first source register are identical, or
- the registers used are the 8 most popular ones.

The C extension is compatible with all other standard instruction extensions. The C extension allows 16-bit instructions to be freely intermixed with 32-bit instructions, with the latter now able to start on any 16-bit boundary.

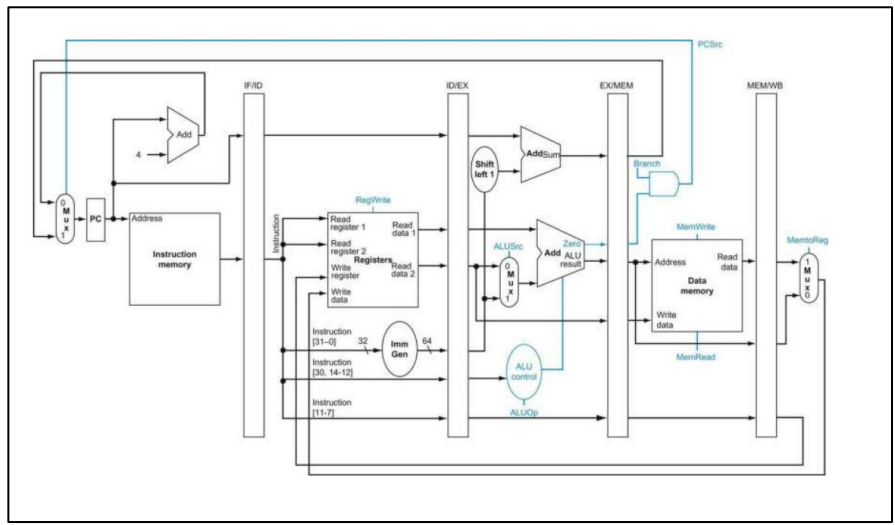
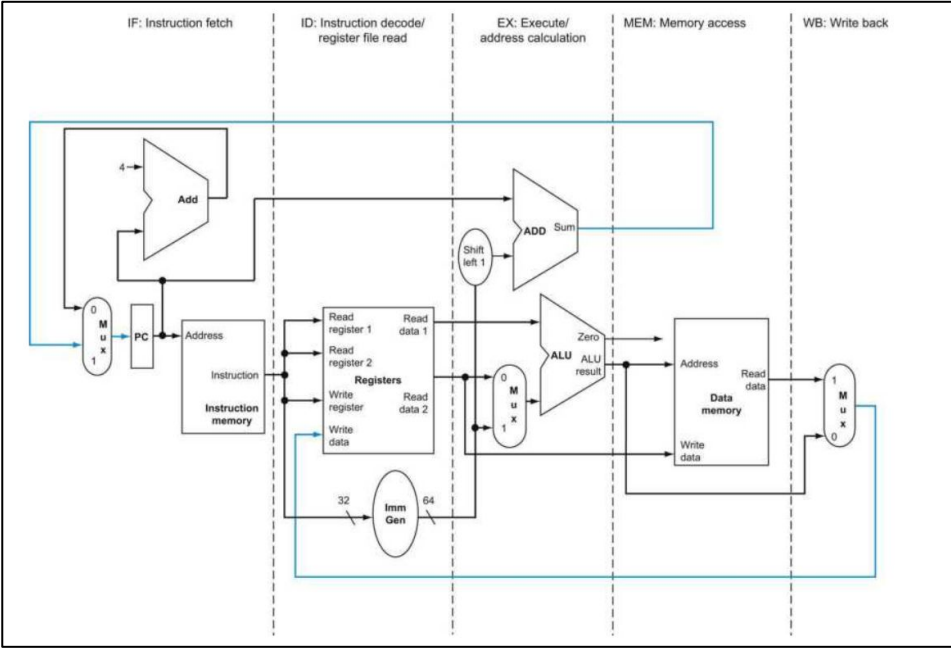
Removing the 32-bit alignment constraint on the original 32-bit instructions allows significantly greater code density.

2. Why have SP-4, SP-8 in code, if there is hardware shift?

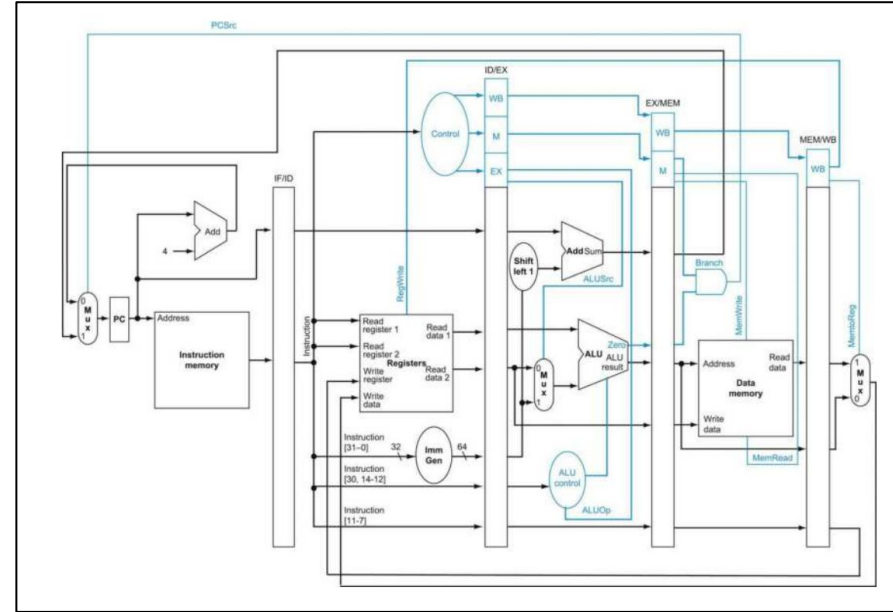
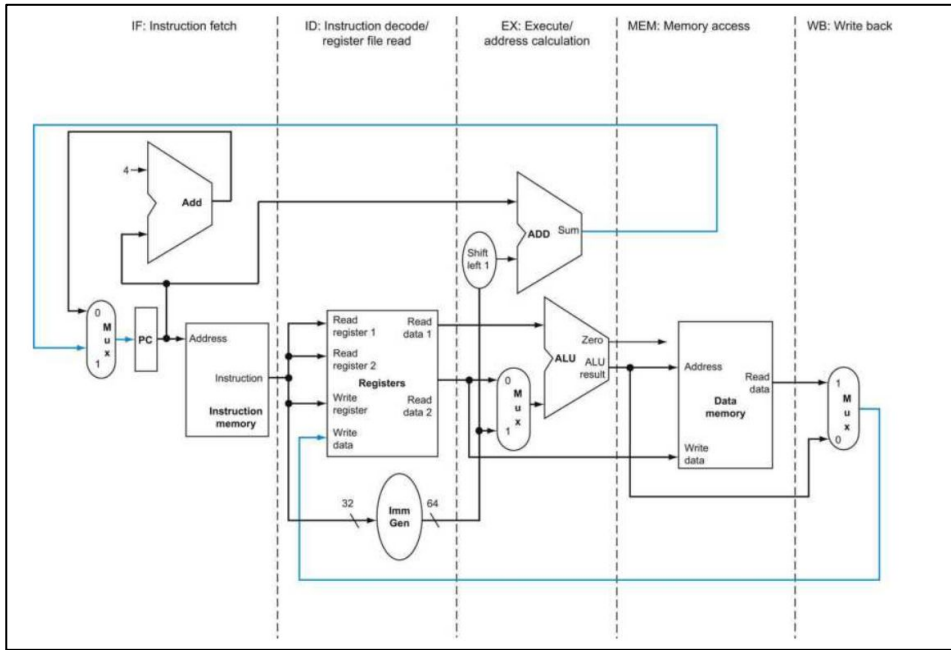
3. Latch in single cycle, non-pipelined vs. registers in pipeline



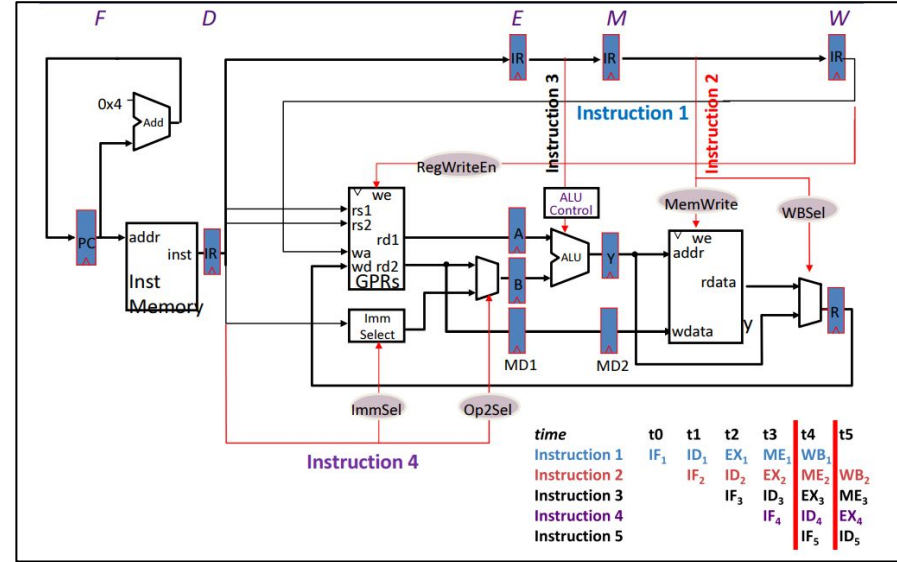
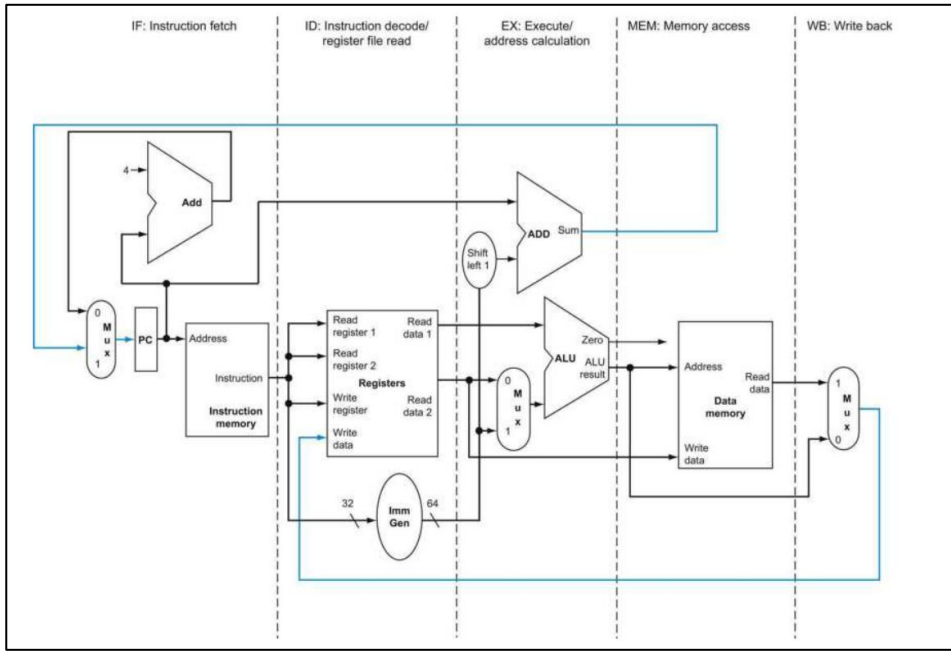
3. Latch in single cycle, non-pipelined vs. registers in pipeline



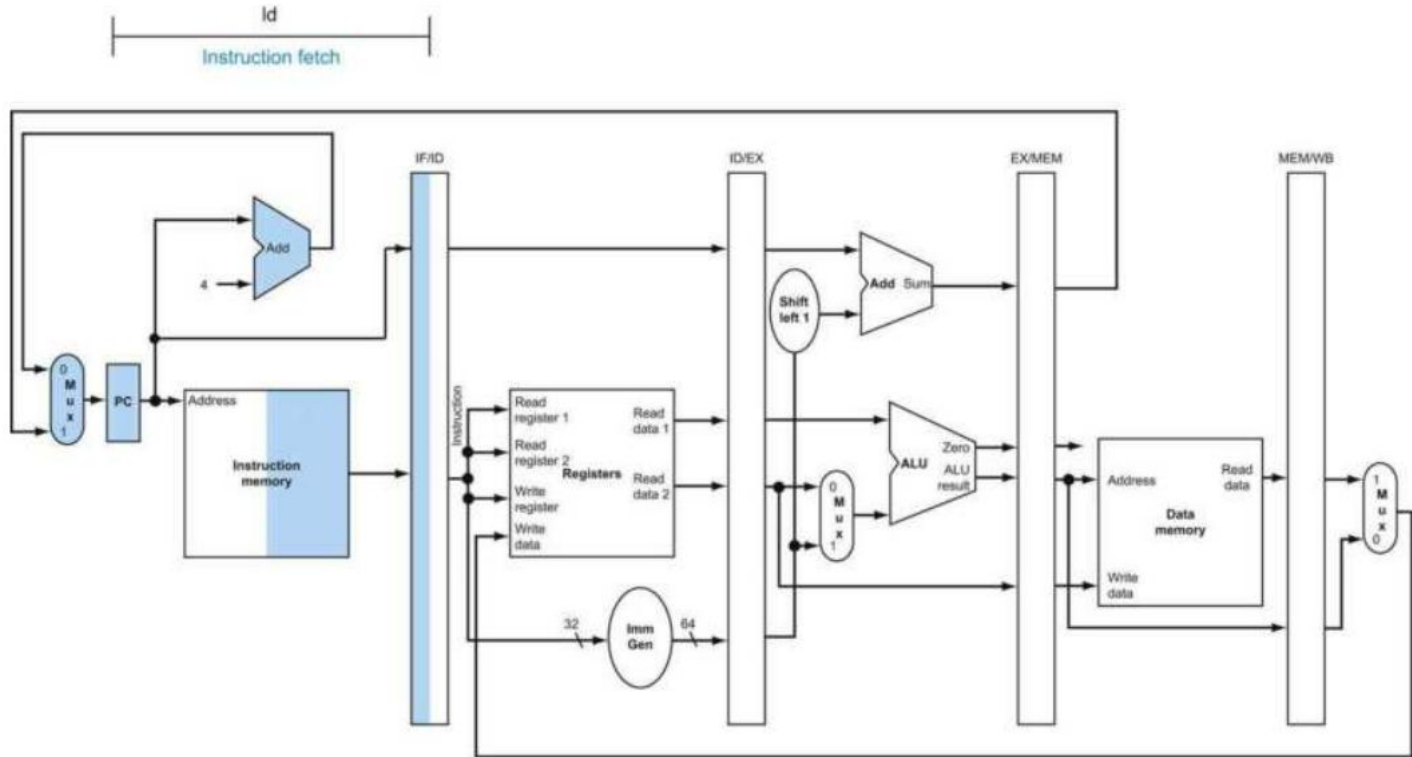
3. Latch in single cycle, non-pipelined vs. registers in pipeline



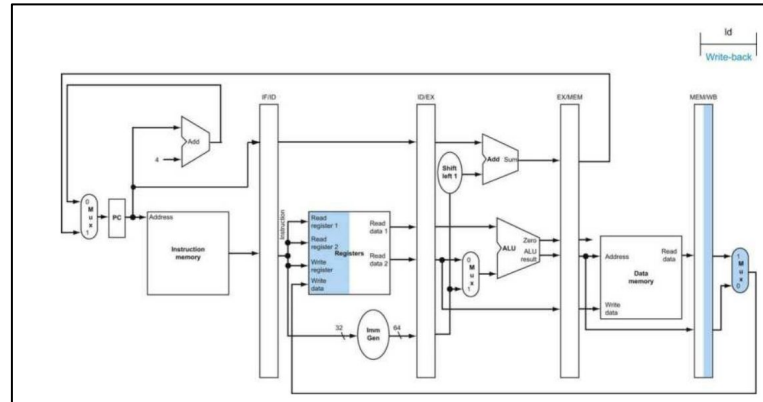
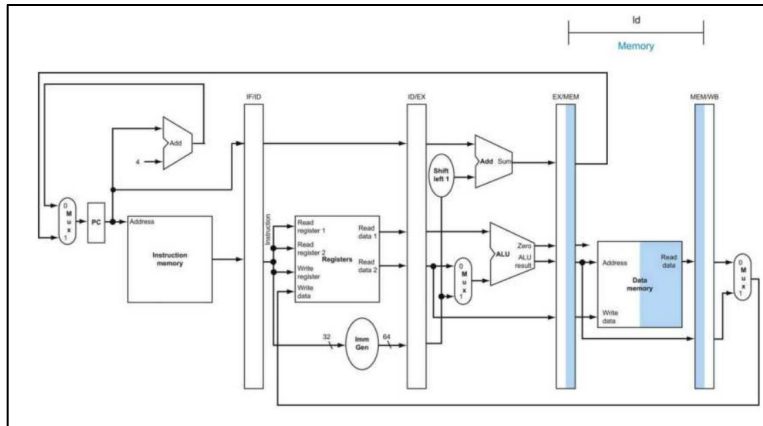
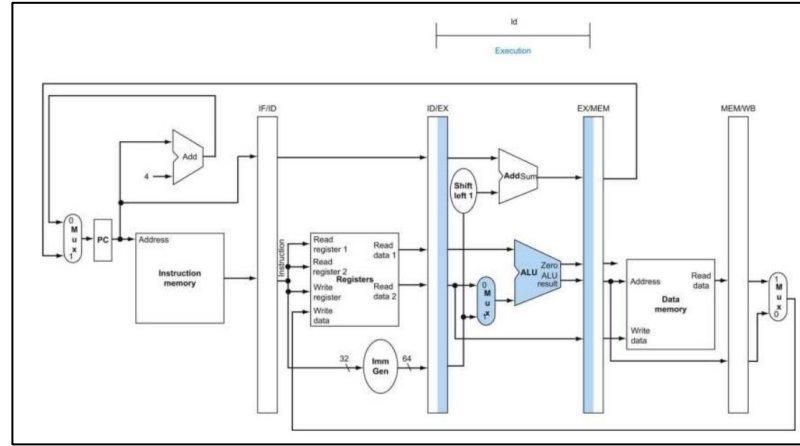
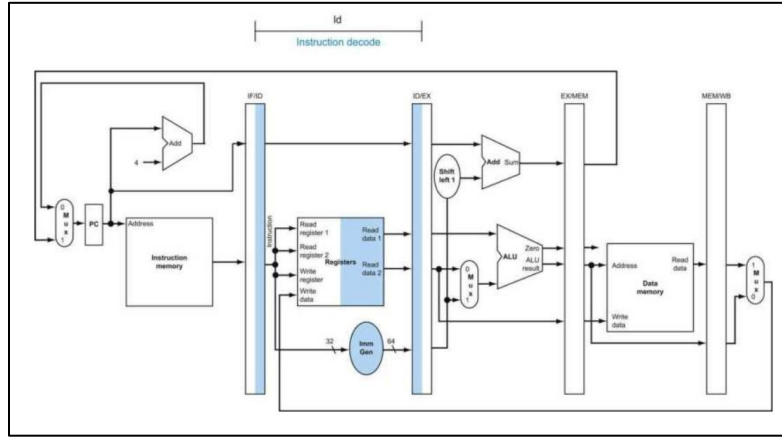
3. Latch in single cycle, non-pipelined vs. registers in pipeline



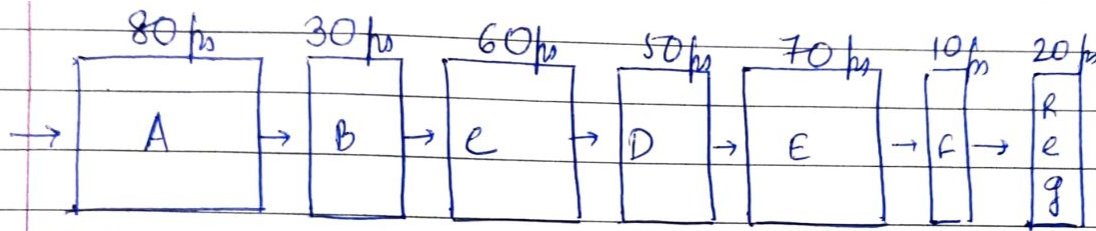
4. Where to add the register delay - stage before or after?



4. Where to add the register delay - stage before or after?

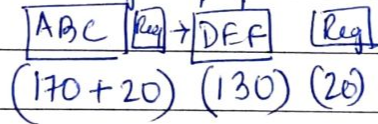


5. Different staged pipelines

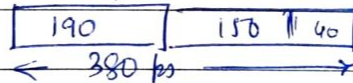


Pipeline register has a delay of 20 ps.

Where do we use instead a single pipeline register to have a 2 stage pipeline?



latency



latency

380 ps

2stage pipeline

(B)

throughput

$$\frac{190 \times 10^{-12} \text{ pico second}}{1 \text{ second}}$$

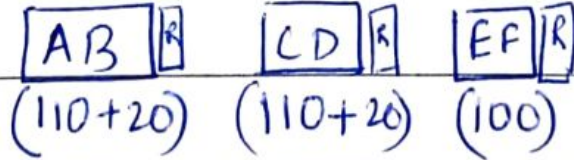
1 instruction

$$\frac{10^{12}}{190} = \frac{100 \times 10^9}{19}$$

$$= 5.26 \text{ GIPS.}$$

5. Different staged pipelines

3 stage pipeline



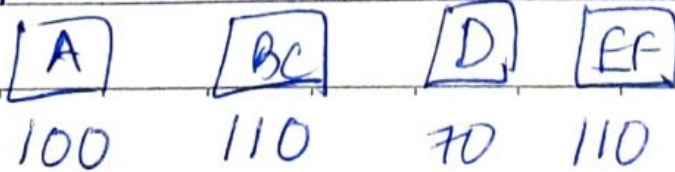
130ps

130ps

390 ps latency

$$\frac{10^{12}}{130} = \frac{100}{13} = 7.6$$

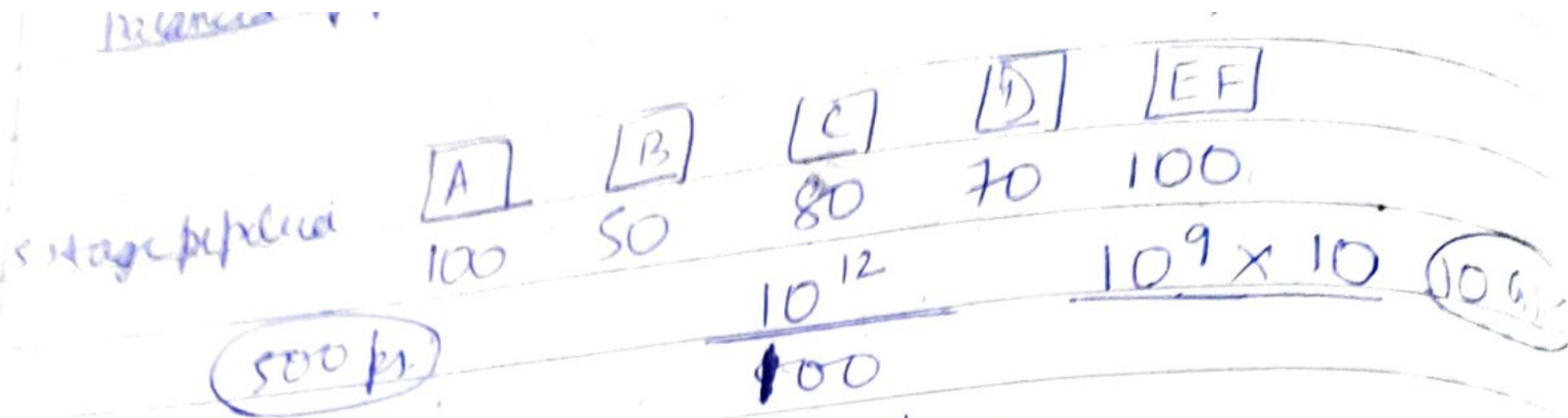
4 stage pipeline



440ps

$$\frac{100}{11} = 9.09$$

5. Different staged pipelines



bounded by the slowest stage

6. Loop code pipelined - initial value in R4 is 400

loop: LW R1, 0 (R4) ; Load first operand

LW R2, 400 (R4) ; Load second operand

ADDI R3, R1, R2 ; Add operands

SW R3, 0 (R4) ; Store result

SUB R4, R4, #4 ; Calculate address of next element

BNEZ R4, L1 ; Loop if (R4) != 0

6. Non-pipelined

loop: LW R1, 0 (R4) ; Load first operand

LW R2, 400 (R4) ; Load second operand

ADDI R3, R1, R2 ; Add operands

SW R3, 0 (R4) ; Store result

SUB R4, R4, #4 ; Calculate address of next element

BNEZ R4, L1 ; Loop if (R4) != 0

6. Without forwarding or bypassing

loop: LW R1, 0 (R4) ; Load first operand

LW R2, 400 (R4) ; Load second operand

ADDI R3, R1, R2 ; Add operands

SW R3, 0 (R4) ; Store result

SUB R4, R4, #4 ; Calculate address of next element

BNEZ R4, L1 ; Loop if (R4) != 0

6. With forwarding or bypassing

loop: LW R1, 0 (R4) ; Load first operand

LW R2, 400 (R4) ; Load second operand

ADDI R3, R1, R2 ; Add operands

SW R3, 0 (R4) ; Store result

SUB R4, R4, #4 ; Calculate address of next element

BNEZ R4, L1 ; Loop if (R4) != 0

6. Instruction reordering

loop: LW R1, 0 (R4) ; Load first operand

LW R2, 400 (R4) ; Load second operand

ADDI R3, R1, R2 ; Add operands

SW R3, 0 (R4) ; Store result

SUB R4, R4, #4 ; Calculate address of next element

BNEZ R4, L1 ; Loop if (R4) != 0

loop: LW R1, 0 (R4) ; Load first operand

LW R2, 400 (R4) ; Load second operand

SUB R4, R4, #4 ; Calculate address of next element

ADDI R3, R1, R2 ; Add operands

BNEZ R4, L1 ; Loop if (R4) != 0

SW R3, 0 (R4) ; Store result

Multiplier circuit — the shifts are not shown

Minor syllabus, what we have learnt so far

Minor syllabus, what we have learnt so far