SNS COLLEGE OF ENGINEERING

Kurumbapalayam (PO), Coimbatore – 641 107
Accredited by NAAC-UGC with 'A' Grade
Approved by AICTE, Recognized by UGC & Affiliated to Anna University, Chennai

DEPARTMENT OF INFORMATION TECHNOLOGY

COURSE NAME: 19IT301 COMPUTER ORGANIZATION AND ARCHITECTURE

II YEAR/ III SEM

Unit 3 : Processor and Pipelining

Topic 8: Influence on Instruction sets – Data path and control consideration

Influence on Instruction Sets

Overview

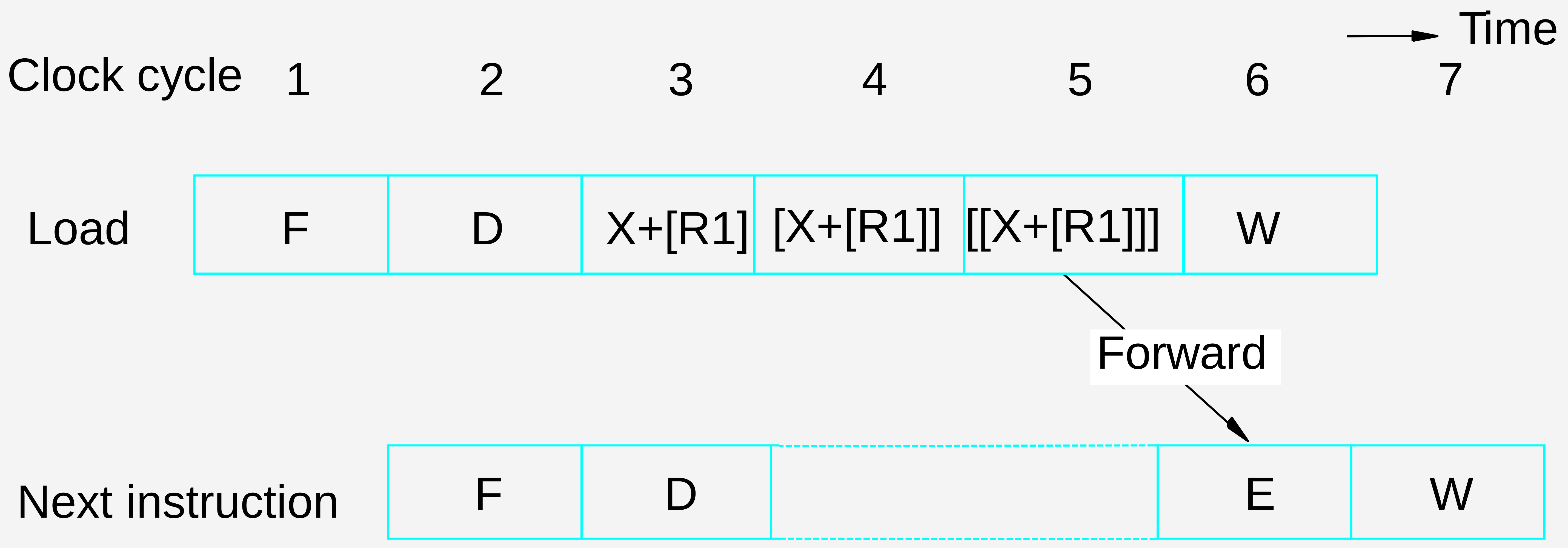
- Some instructions are much better suited to pipeline execution than others.
- Instruction side effects can lead to undesirable data dependencies
- Examining relationship between pipelined execution and machine instruction features
- 2 key aspects of machine instructions
    - Addressing modes
    - Conditional code flags

Addressing Modes

- Addressing modes include simple ones and complex ones.
- In choosing the addressing modes, consider the effect of each addressing mode on instruction flow in the pipeline:
    - Side effects
    - The extent to which complex addressing modes cause the pipeline to stall

Complex Addressing Mode

Load (X(R1)), R2

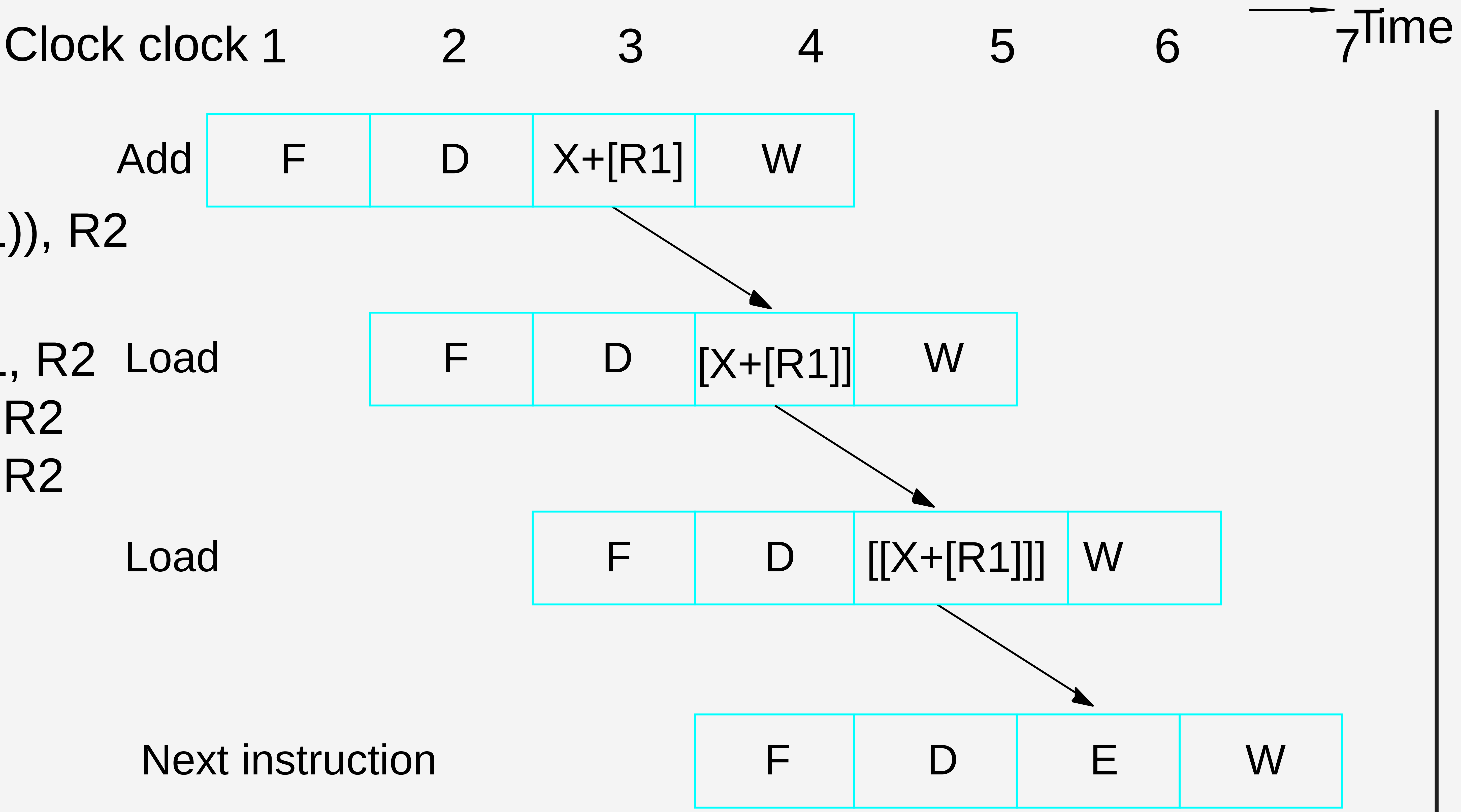


(a) Complex addressing mode

Simple Addressing Mode

Load (X(R1)), R2

Add #X, R1, R2
Load (R2), R2
Load (R2), R2



Addressing Modes

- In a pipelined processor, complex addressing modes do not necessarily lead to faster execution.
- **Advantage:** reducing the number of instructions / program space
- **Disadvantage:** Long execution cause pipeline to stall / more complex hardware to decode / not convenient for compiler to work with
- **Conclusion:** complex addressing modes are not suitable for pipelined execution.
    - Good addressing modes should have the features:
        - Access to an operand does not require more than one access to the memory
        - Only load and store instruction access memory operands
        - The addressing modes used do not have side effects
    - Register, register indirect, index have these above features

Conditional Codes

- If an optimizing compiler attempts to reorder instruction to avoid stalling the pipeline when branches or data dependencies occur, it must ensure that reordering does not cause a change in the outcome of a computation.
- The dependency introduced by the condition-code flags reduces the flexibility available for the compiler to reorder instructions.

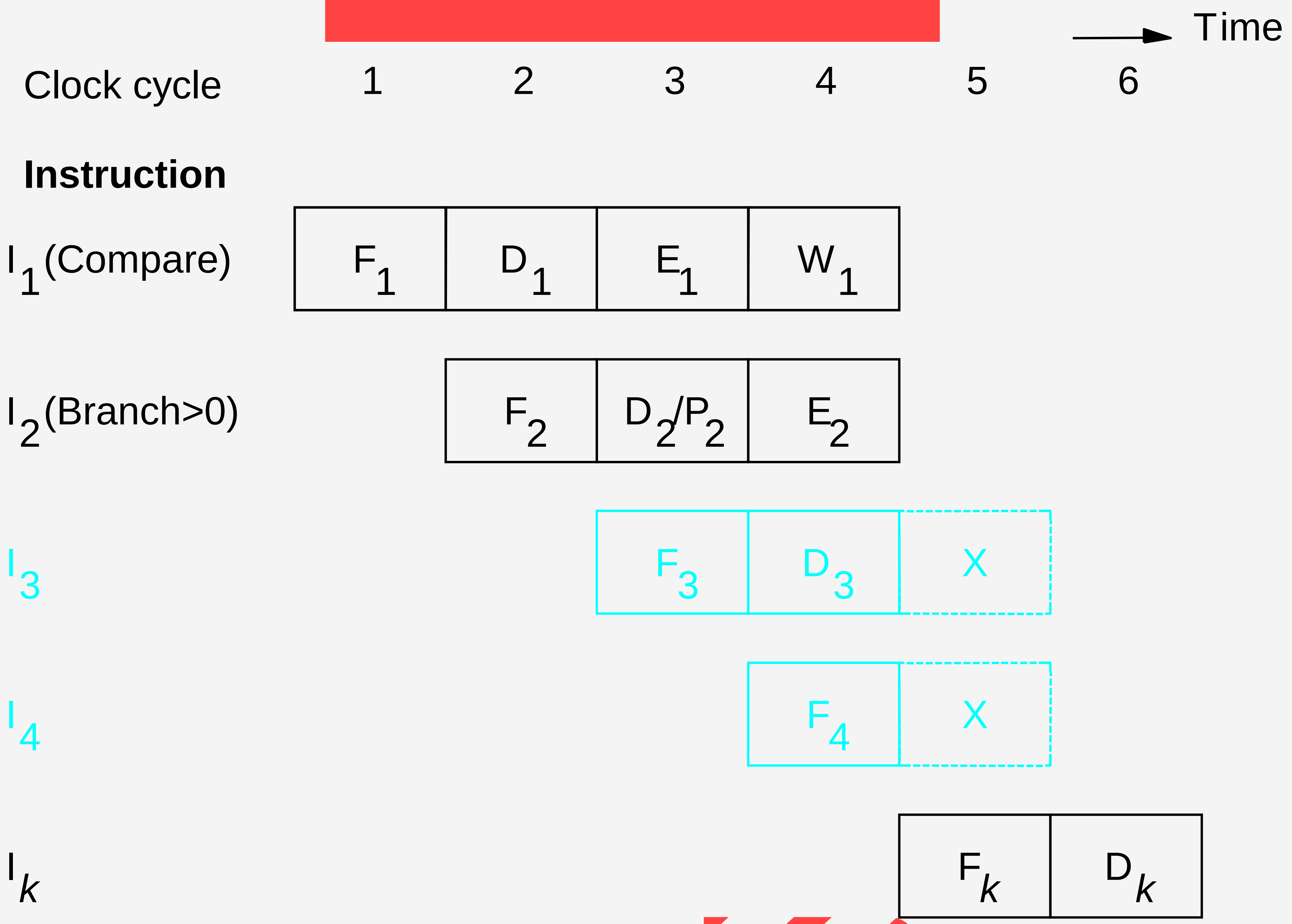
Conditional Codes

```
Add           R1,R2
Compare       R3,R4
Branch=0      . . .
```

(a) A program fragment

```
Compare       R3,R4
Add           R1,R2
Branch=0      . . .
```

(b) Instructions reordered



Instruction reordering

Conditional Codes

Two conclusion:

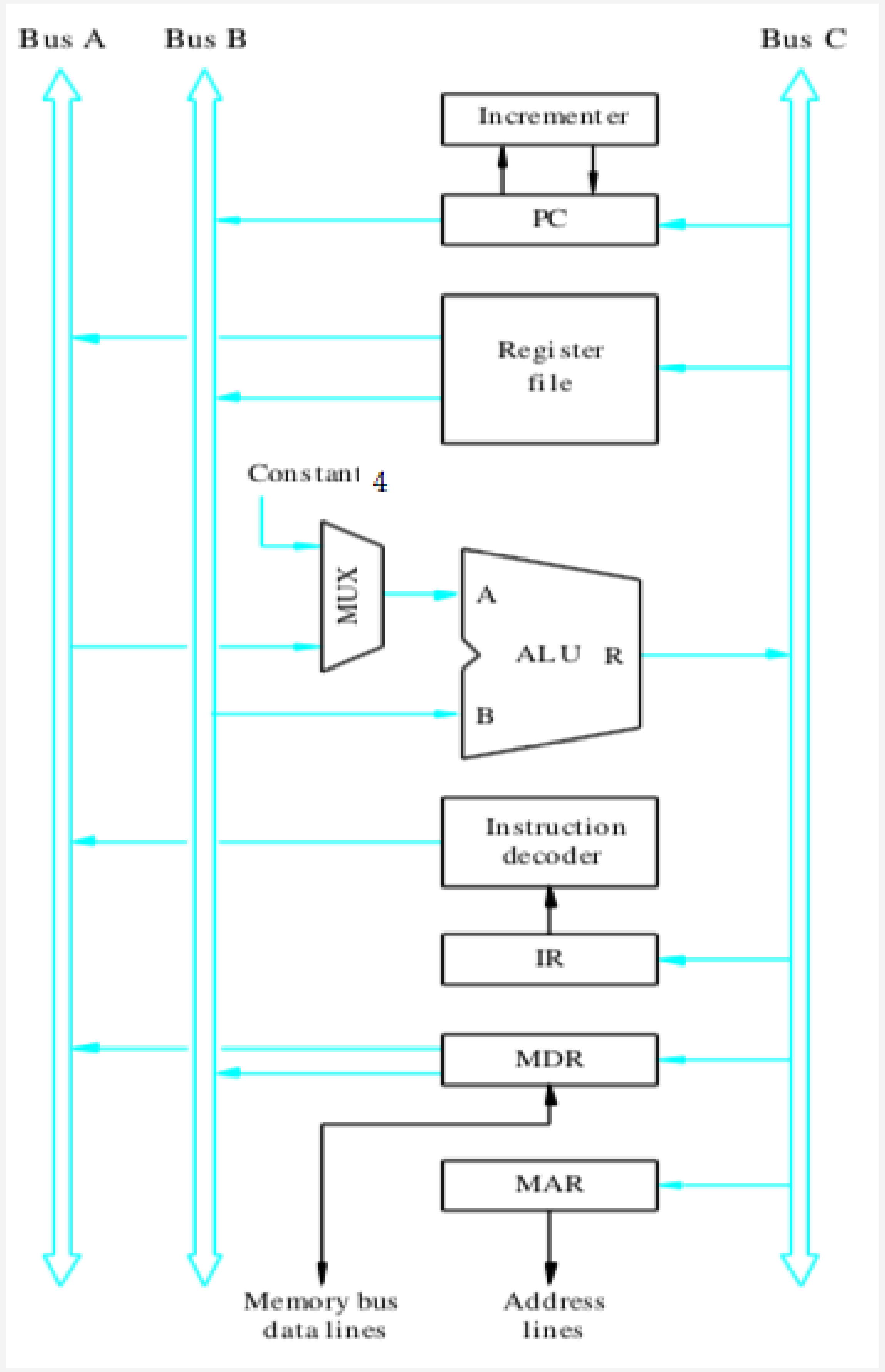
➢To provide flexibility in reordering instructions, the condition-code flags should be affected by as few instruction as possible.

➢The compiler should be able to specify in which instructions of a program the condition codes are affected and in which they are not.

Datapath and Control Considerations

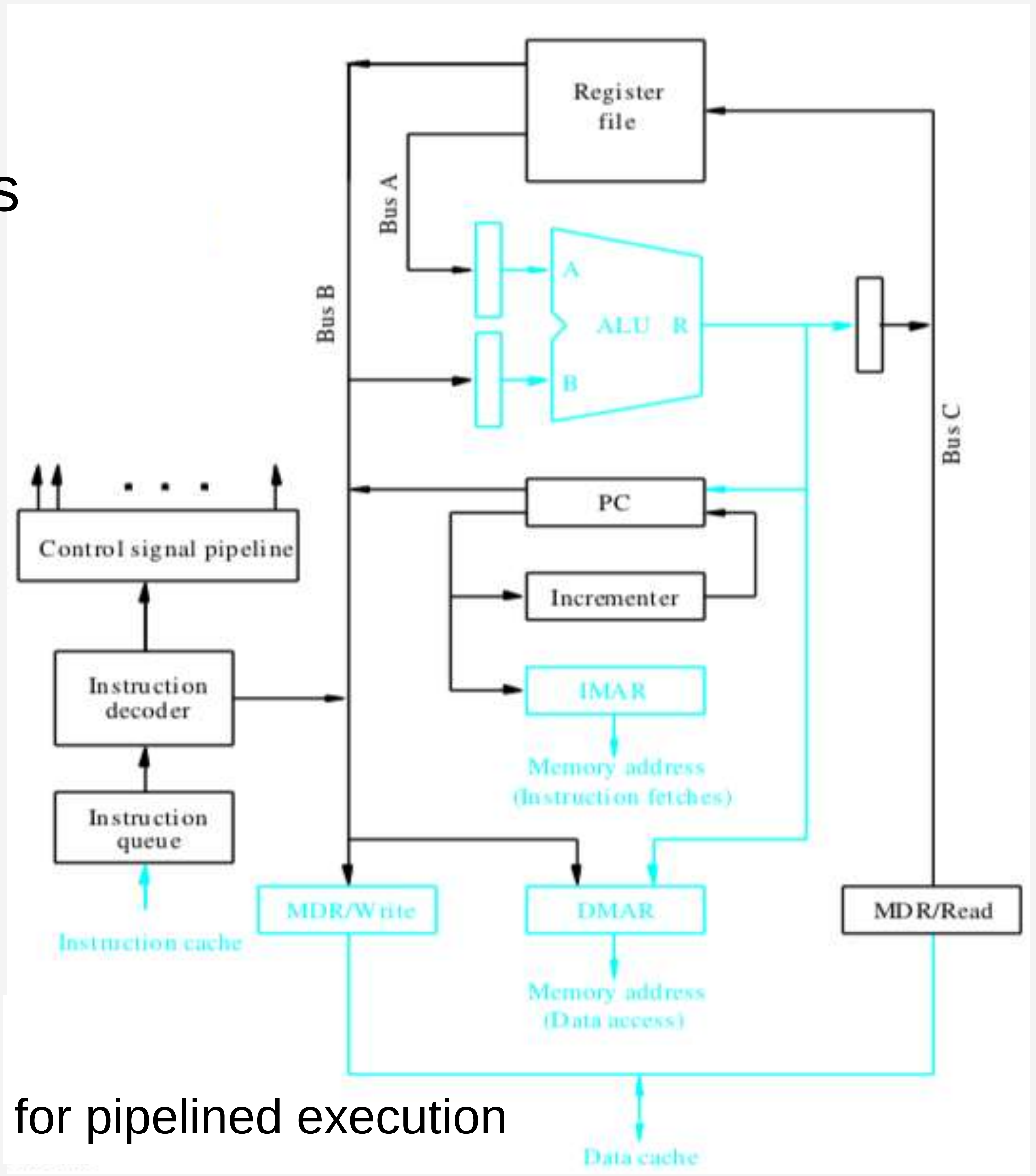
Original Design

3 bus organization of the
datapath



Pipelined Design

- Separate instruction and data caches
- PC is connected to IMAR
- DMAR from register or from ALU
- Separate MDR for read and write
- Buffers for ALU
- Instruction queue
- Instruction decoder output



Datapath modified for pipelined execution

Pipelined Design

Independent operations:

- Reading an instruction from the instruction cache
- Incrementing the PC
- Decoding an instruction
- Reading from or writing into the data cache
- Reading the contents of up to two regs
- Writing into one register in the reg file
- Performing an ALU operation

Thank You