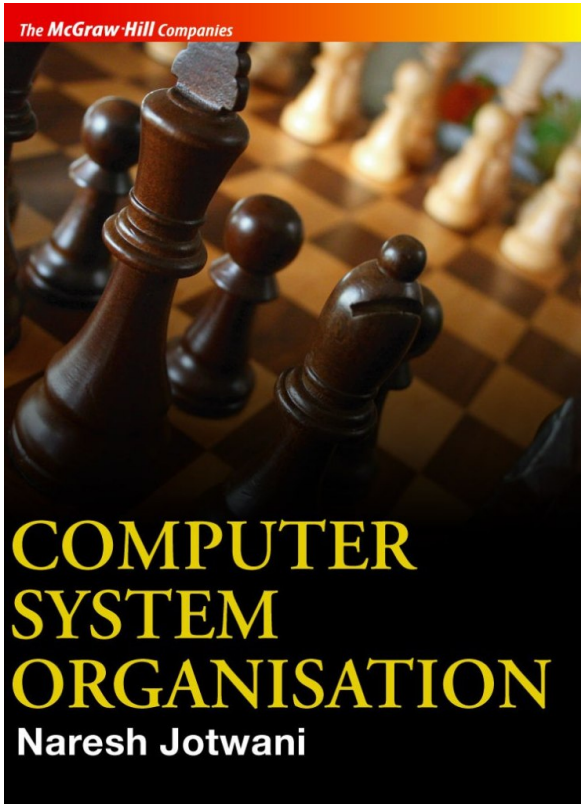
Mc Graw Hill Education

COMPUTER SYSTEM ORGANISATION

Naresh Jotwani

PowerPoint Slides



Chapter 6

PROCESSOR DESIGN

Introduction:

Internal design of the processor can be separated into two parts:

- (i) *Processor architecture* – defined by registers, functional units, and data paths
- (ii) *Control unit* – mechanism for generating the right control signals at the right time instants

Processor Architecture

- *Arithmetic and logic unit* (ALU) - For carrying out arithmetic and logic operations on data
  - integer addition and subtraction, bit-wise logical operations, shift operations, etc.
- *Floating point unit* (FPU) - For floating point operations (though not all processors have FPU)
- *Instruction register* (IR) - For storing the instruction currently under execution
- *Program counter* (PC) - stores memory address of the next instruction to be fetched

- *Program status word* (PSW) - register which contains flag bits Z, N, O, C and P, and some other useful information about the running program

- *Address* and *Data* – registers for interacting with main memory and input/output device controllers

  – They hold, respectively, the required address and data involved in read/ write operation with main memory or I/O.

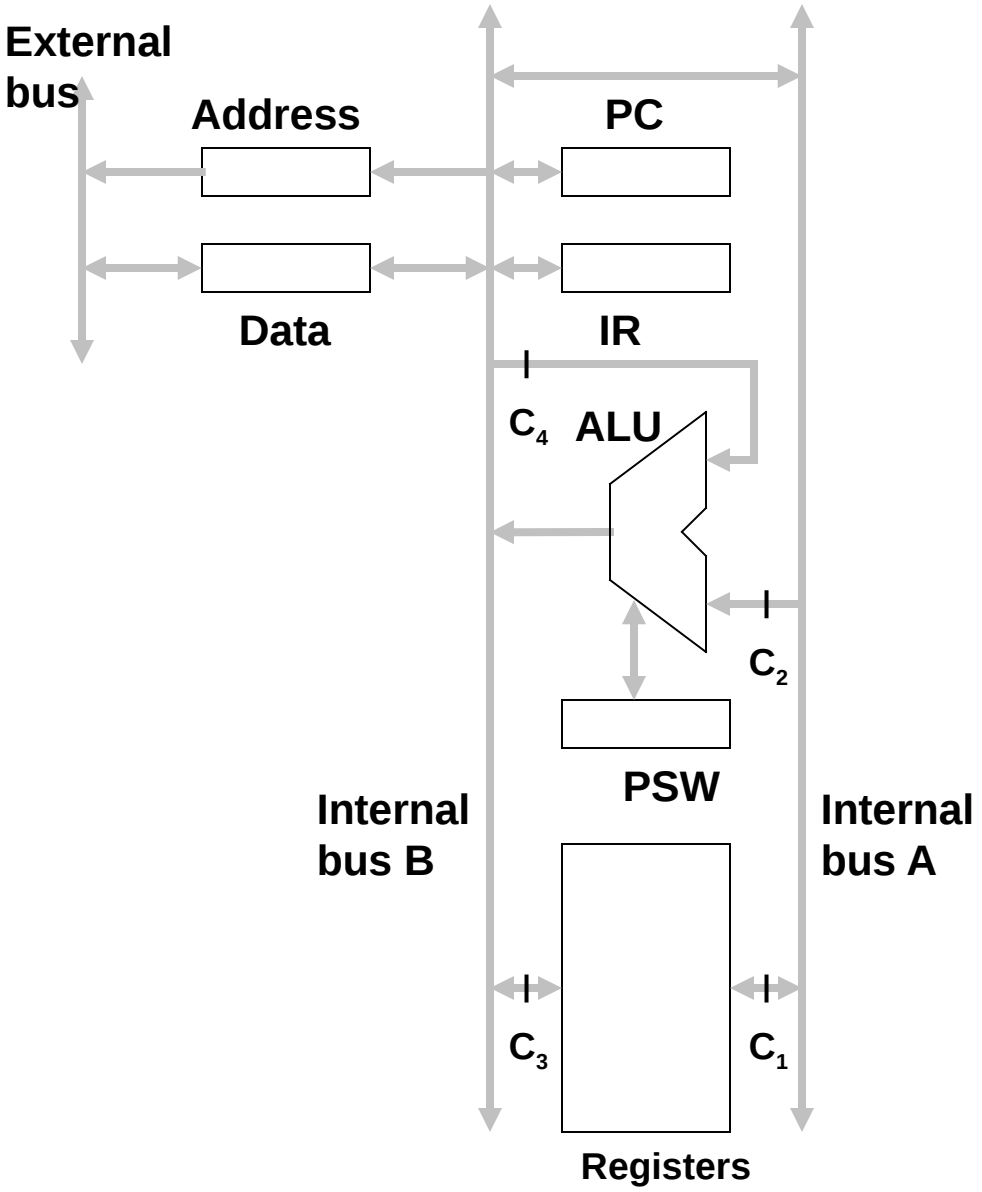
- Thus IR, PC, PSW, Address & Data are special purpose registers

- Internal *data paths* / *data busses* – Equal in width to the word size of the processor, to connect various elements of the processor

- Processor also has an *external bus* connecting other elements of the system

- A*ddress* and *Data* registers are connected respectively to *address* and *data lines* of the external bus, which also has *control signals* to govern bus data transfers

Architecture of a simple processor:



Control signals and control unit:

- Processor operations are determined by *control signals* applied to processor elements.

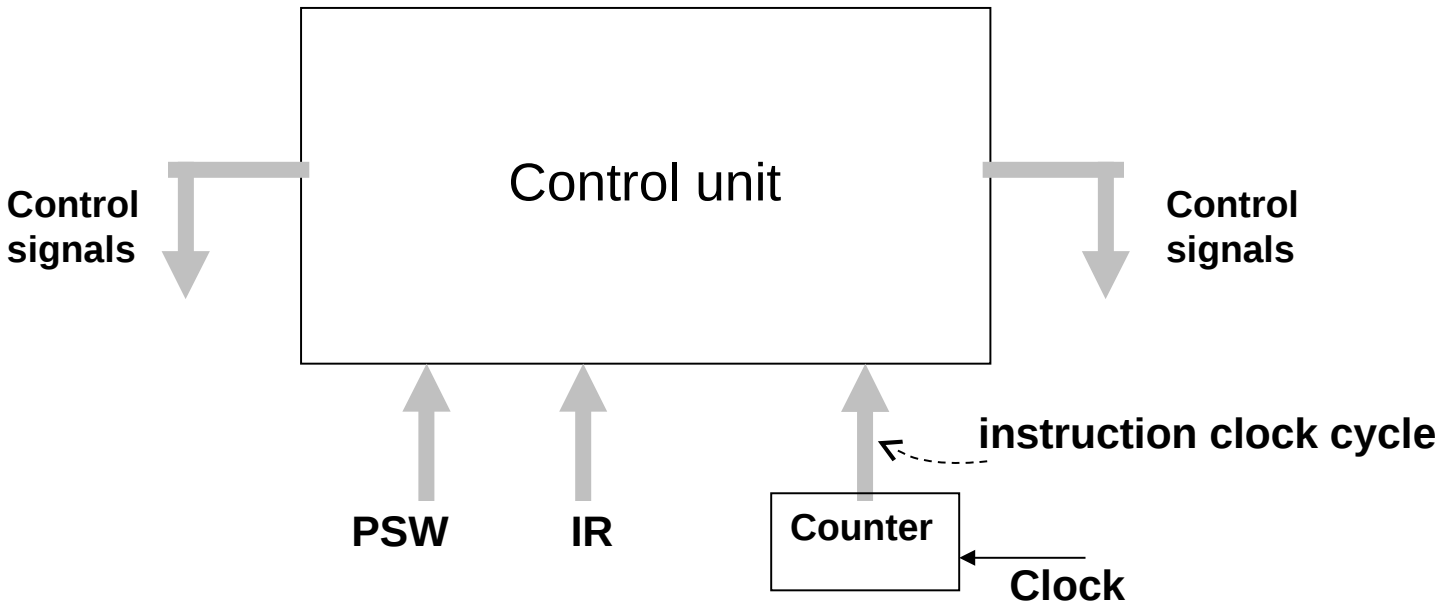
- Four illustrative points in the processor (previous slide) where control signals may be applied to carry out specific operations – C_1, C_2, C_3 and C_4
  - With appropriate combination of these control signals, data from a selected register is supplied as one of the two inputs to ALU, via internal busses A and/or B.

- Data transfers between registers and ALU can take place *in parallel* if the required hardware elements along the two respective paths are independent of each other.

- Such parallelism is an important goal to achieve in processor architecture.

- Even a relatively simple architecture typically requires a few dozen control signals for its proper operation.

Control unit of the processor generates correct control signals in each processor clock cycle



A typical Control Unit

- Inputs to the Control Unit:

  IR, PSW, instruction clock cycle

- Say an instruction takes *n* clock cycles to fetch and execute; these cycles may be numbered from 0 to *n*-1.

- Control signals generated depend on the clock cycle within the instruction; the counter shown counts from 0 to *n*-1.

- Note that the *counter* (on the lower right) has a clock signal connected to it.

- This counter tracks the processor clock cycle – from 0 to *n*-1 – in the combined fetch & execute phases of each instruction.

- The counter is reset to zero when a new fetch phase starts.

- The Control Unit circuit may be quite complex due to its many inputs and outputs. But, in principle, it is a combinational circuit.

- Control Unit designed and built as one combinational circuit is known as *hardwired* control unit.

- Once built, such a control unit cannot be easily changed.

- An alternative technique for building the control unit is known as *microprogrammed* control.

Instruction formats and codes:

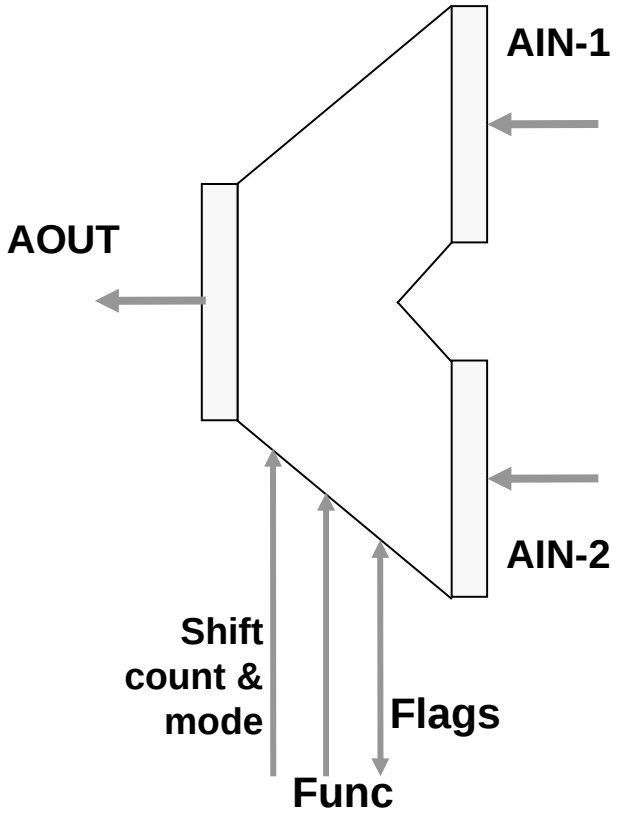
- Processor instructions, instruction modifiers, and addressing modes must be coded into binary for the processor circuits to execute them.

- Possible design objectives in coding an instruction set into binary:
  - To generate compact machine language program
  - To minimize gate delays within the control unit
  - To minimize number of gates in control circuit

- The first two of these would have a direct bearing on the speed of execution of a program.

Arithmetic & Logic Unit (ALU):

- ALU of the processor performs integer addition and subtraction operations, logic operations, and shift operations.
- Operations can be described as combinational functions of inputs.
- ALU is a combinational circuit which must be designed to reduce total gate delay, so that the processor can operate at higher clock speed.


AIN-1
AOUT
AIN-2
Shift
count &
mode
Flags
Func

- The function code supplied to the ALU is shown at the bottom, with label ‘Func’
- Control input to the *shifter* part of ALU is shown with the label ‘Shift count & mode’
- Bi-directional arrow with label ‘Flags’ shows the relationship of the conditional flags to ALU operations
- ALU can be connected to the internal busses even without the input and output registers shown.
- In modern processors having multiple FPUs / ALUs, each such functional unit needs its own input and output registers.

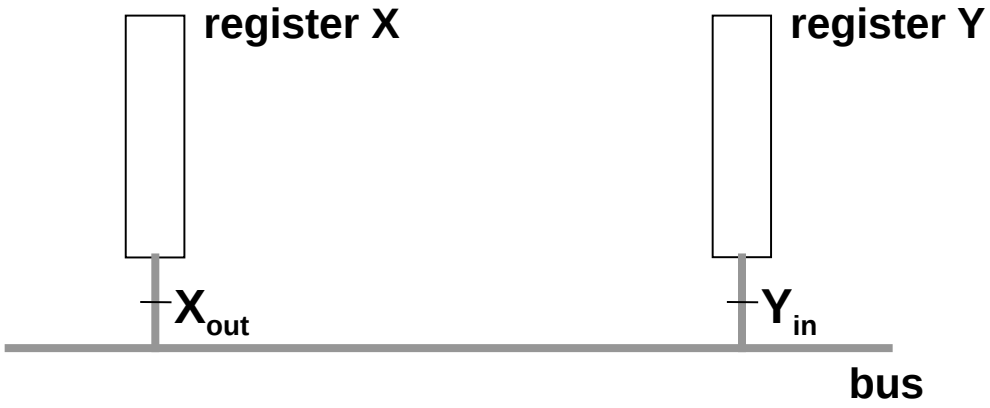
Data transfer and manipulation:

- Control signals perform individual *micro-operations* within the processor.
- Function specified by a machine instruction is carried out through the right combination of *micro-operations* in each instruction clock cycle.
- In a functional unit, any data *manipulation* – i.e. addition, subtraction, shift, and so on – occurs only when the required control signals are applied to the unit.
- ALU and FPU are examples of such functional units.

- Figure shows how contents of register X are transferred to register Y over a data bus.

- To place its contents on the bus, register X is provided with control signal X_{out}; and to gate the data in from the bus, register Y is provided with control signal Y_{in}.

- In a clock cycle, if both X_{out} and Y_{in} are asserted, the result is that the contents of register X will get copied into register Y over the bus.



Program counter (PC):

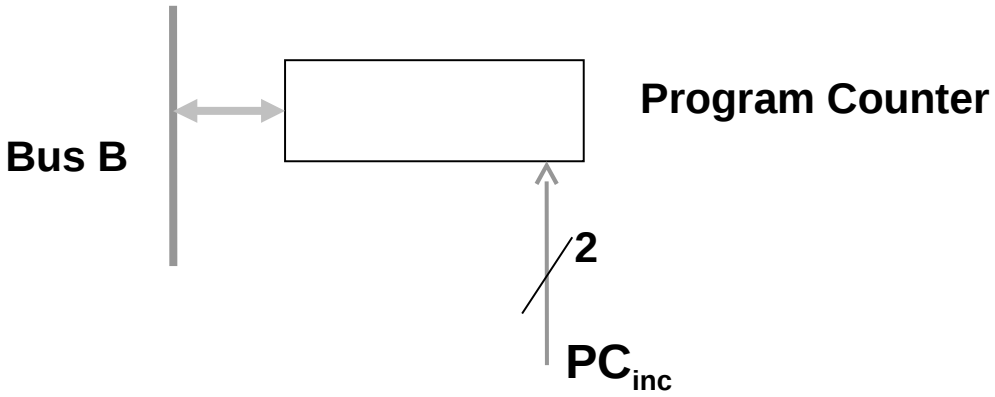
- A special register in the processor which contains the memory address of the next instruction to be fetched and executed.

- At the end of execution of one instruction, the processor executes the next instruction, which is one of the following:
  - (a) the immediate next instruction stored in memory,
  - (b) after a jump or a function call, the instruction stored at the target address, or
  - (c) after a return instruction, the instruction after the last function call (recall Chapter 5).

- On *NICE*, machine instructions have length of one, two or four bytes. There is a 2-bit wide control signal PC_{inc} associated with the PC.

- The three possible non-zero values of PC_{inc} – which are 01, 10 and 11 – cause PC to be incremented by 1, 2 or 4, respectively.

- The required circuit for incrementing PC is provided with it, so the ALU is not used for this purpose.



Nested function calls and the Stack:

- For function call and return instructions CALL and RET, there is a need to store return addresses of *nested* function calls .

- Recall that the last function called is the first one from which return is made.

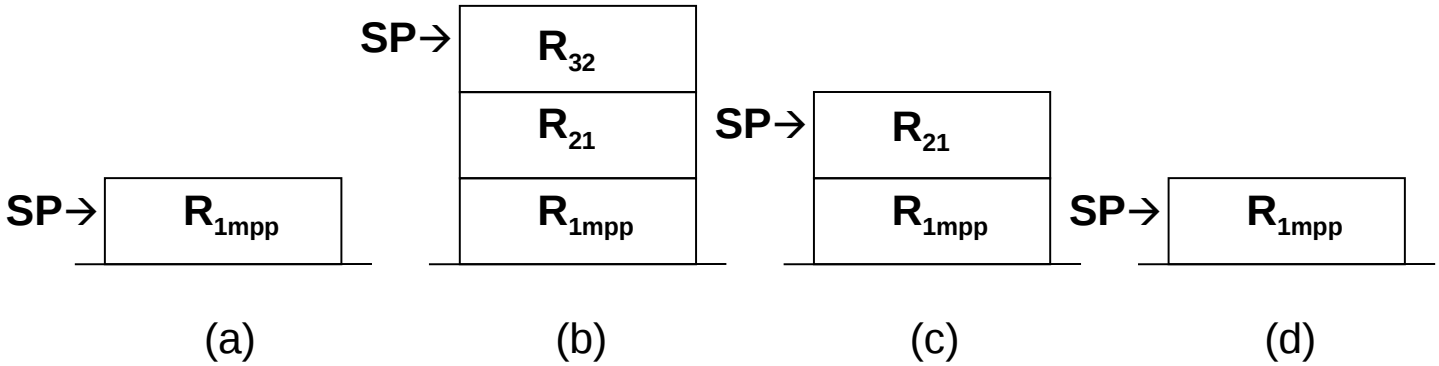
- Also, we cannot place an arbitrary limit on the *depth of nesting*.

- We use a mechanism known as *stack*, which resides in main memory.

- In the figure (next slide), the three return addresses are denoted by R_{1mp}, R_{21} and R_{32}, which is the order in which we store them as the main program calls function F_1, F_1 calls F_2, and F_2 calls F_3.

- 'SP→' indicates that register SP in the processor holds the address of the last element added to the *stack*

- No matter how many elements are added to the stack, only one *stack pointer* register SP is needed to hold the address of *top of stack*.


SP→
R_{1mpp}
SP→
R_{32}
R_{21}
R_{1mpp}
SP→
R_{21}
R_{1mpp}
SP→
R_{1mpp}
(a)
(b)
(c)
(d)

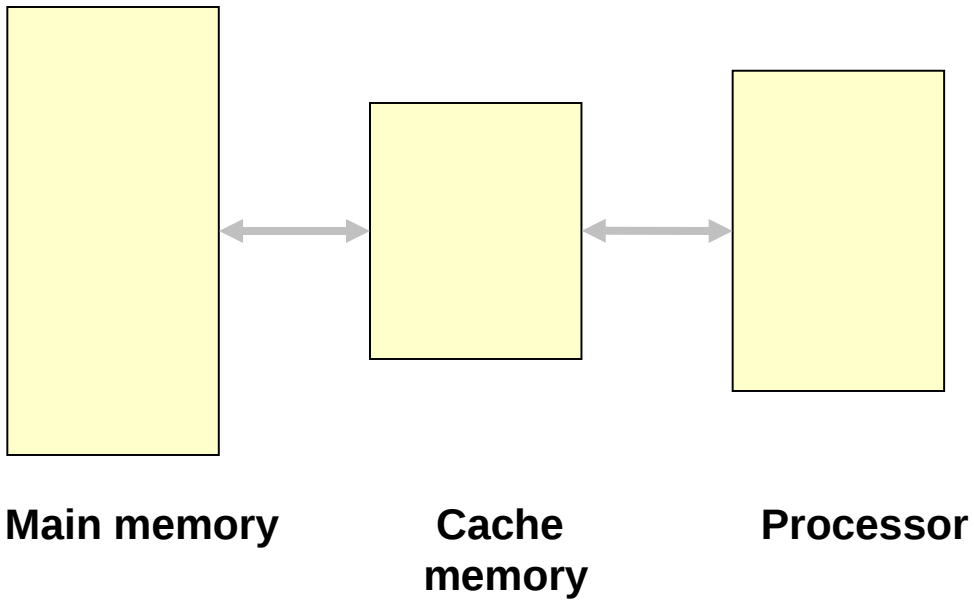
- When an element is added to the stack, with its consequent change in SP, the operation is known as a PUSH operation.

- When an element is removed from the stack, with the opposite change in SP, that operation is known as a POP operation.

- Thus the two main operations on a stack are PUSH and POP.

Cache memory:

Provided to overcome speed mismatch between processor and main memory. Speed of cache memory is much closer to that of the processor.



Effective memory access time =

T_{cache}*cache hit rate + T_{memory}*cache miss rate

For example, assume:

Clock cycle = T_{cache} = 2 ns

T_{memory} = 50 ns

Cache hit rate = 0.9

Then, effective memory access time =

2*0.9 + 50*0.1 = 6.8 ns

- When the processor finds a required memory byte or word in cache, the event is known as a *cache hit*.

- Occasionally the processor will access a memory byte or word not present in the cache. This event is detected as a *cache miss*, and access is made to main memory for the required byte or word.

- In many modern computers, two levels of cache are provided – known as L1 and L2. L1 is closer to the processor. In some systems, L3 cache is also provided, closer to main memory.

- L1 cache is smaller but faster than L2, and L2 cache is smaller but faster than L3.

To achieve the desired cache hit rate, cache memory design makes use of the following two properties of running programs:

1. *Temporal locality* of program references -

   References are more likely to main memory locations recently referenced.

2. *Spatial locality* of program references -

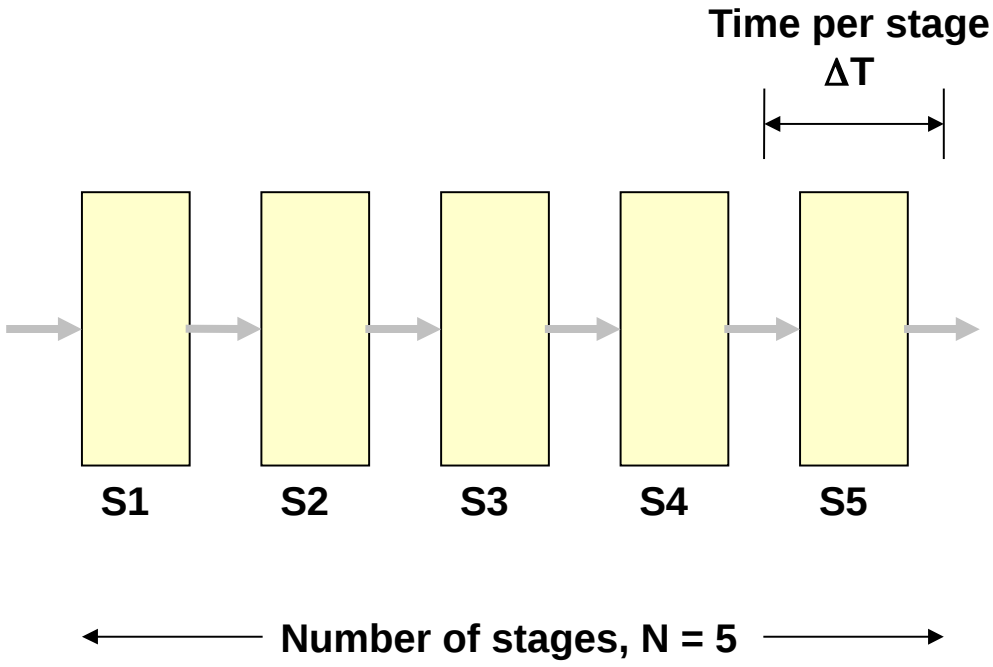
   References are more likely to memory addresses which are close to those recently referenced.

Pipelined implementation:

- Stages of processing a typical instruction are:

1. Instruction fetch (IF) instruction is fetched from cache
2. Instruction decode (ID) fields of instruction in IR are decoded
3. ALU operation (OP) specified ALU operation takes place
4. Memory access (MA) access is made to cache for read/ write
5. Result stored (RS) result is stored in destination register

- In a pipelined implementation, in every processor clock cycle, the instruction moves forward one stage in the pipeline.
- Thus one instruction completes its execution in every clock cycle, while each instruction still does take five clock cycles to complete.
- If an instruction does not require any operation in a particular stage, then it simply passes through that stage as a 'no operation'.



A pipeline of five processing stages

- Access to slow main memory starves a pipelined processor of instructions and data - i.e. a pipelined processor must also be provided with cache memory.

- To avoid contention between pipeline stages for cache access, separate *instruction cache* and *data cache* are usually provided with pipelined processors.

- *Instruction latency* - Total time taken by an instruction through the pipeline stages.
  Processing throughput - Rate of execution of instructions.

- Example: With clock cycle of 2 ns and five stage pipeline, instruction latency is 10 ns, and throughput is 500 million instructions per second.

- Figure in the next slide shows a 5 stage instruction pipeline, and a sequence of instructions, denoted here as i_1, i_2, i_3, and so on.

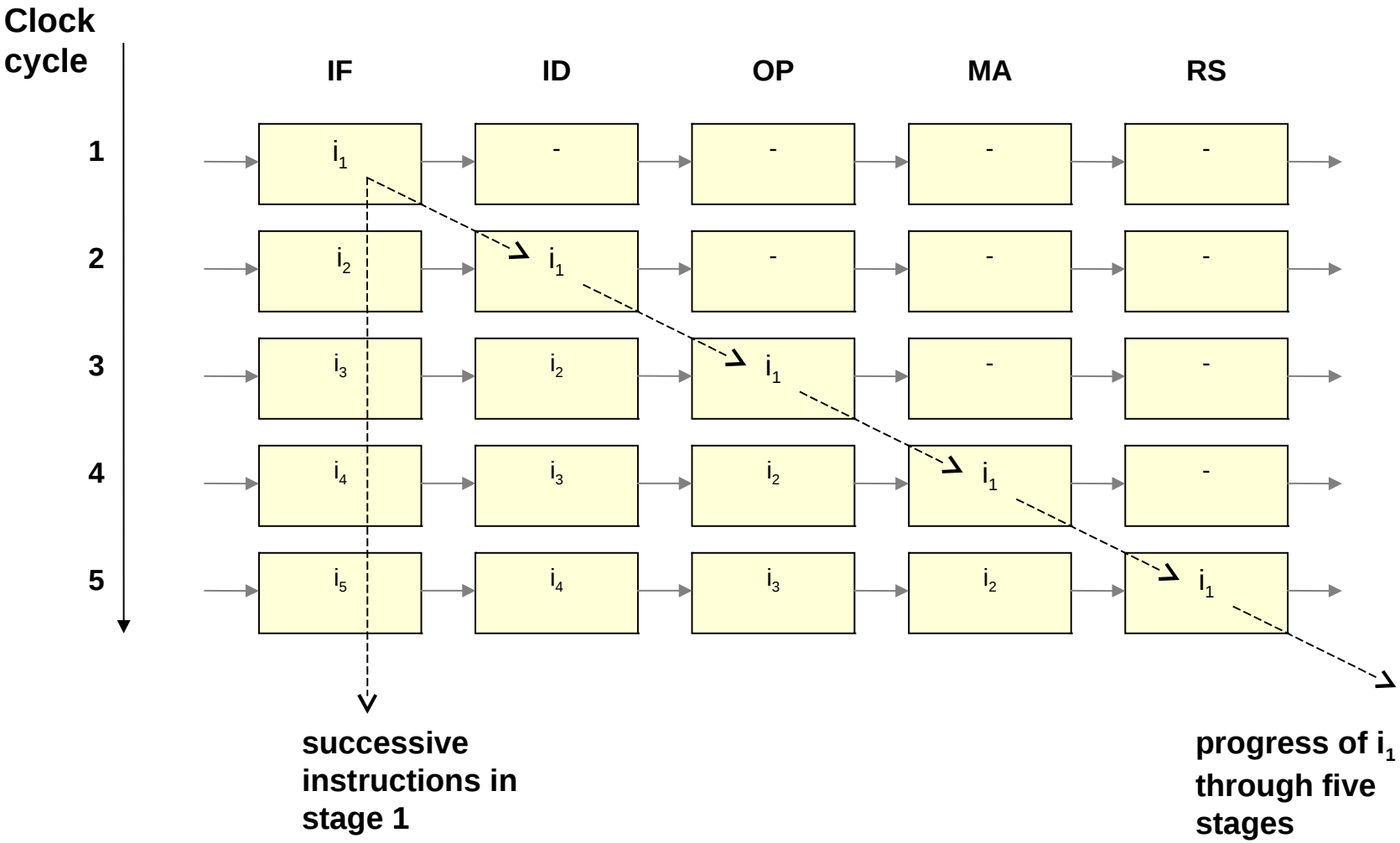
- It is assumed that pipeline is empty before instruction i_1 enters the first stage of instruction fetch (IF).

- The five rows in this figure represent the pipeline in five consecutive clock cycles, as indicated on the left.

- The two dashed arrows drawn in the figure indicate, respectively:

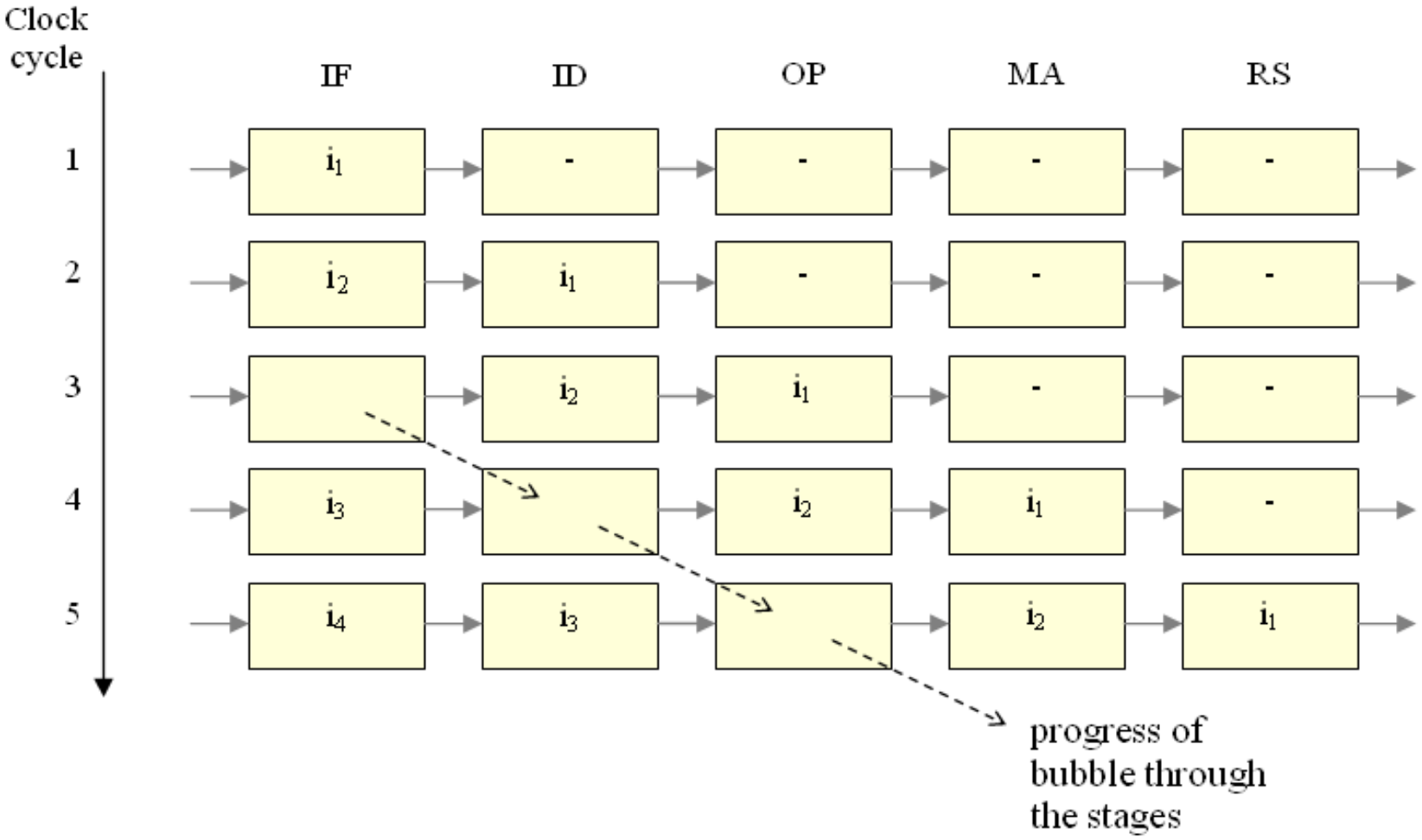
  - (a) the successive occupants of stage 1 (IF), and
  - (b) the progress of i_1 through the five stages.

Instruction pipeline filling up in five clock cycles



- The next slide shows a condition in a pipelined processor which causes it to ‘skip’ one clock cycle in introducing an instruction into the first stage.
  - No instruction enters the first IF stage of the pipeline in clock cycle 3. This stage therefore remains empty in clock cycle 3.
  - When instruction i_3 enters the pipeline in clock cycle 4, the second stage of the pipeline becomes empty, because the instruction i_2 which was in it has moved forward.
- In this way, in every clock cycle, the ‘bubble’ in the pipeline − i.e. a stage which is doing nothing for one clock cycle − moved forward.

A 'bubble' in instruction pipeline



- Conditions which can cause such a ‘bubble’ to enter the pipeline include:

  (i) a cache miss, and

  (ii) a dependency between instructions - i.e. the result of one instruction is not available in time for the subsequent instruction to execute.

- The occurrence of an empty stage in the pipeline is known as a pipeline *stall*. The conditions which cause stalls are known collectively as *hazards*.

- When a transfer of control occurs during program execution, the instruction pipeline must be *flushed*.

- ‘One instruction per processor clock cycle’ is the target which processor designers can aim for.

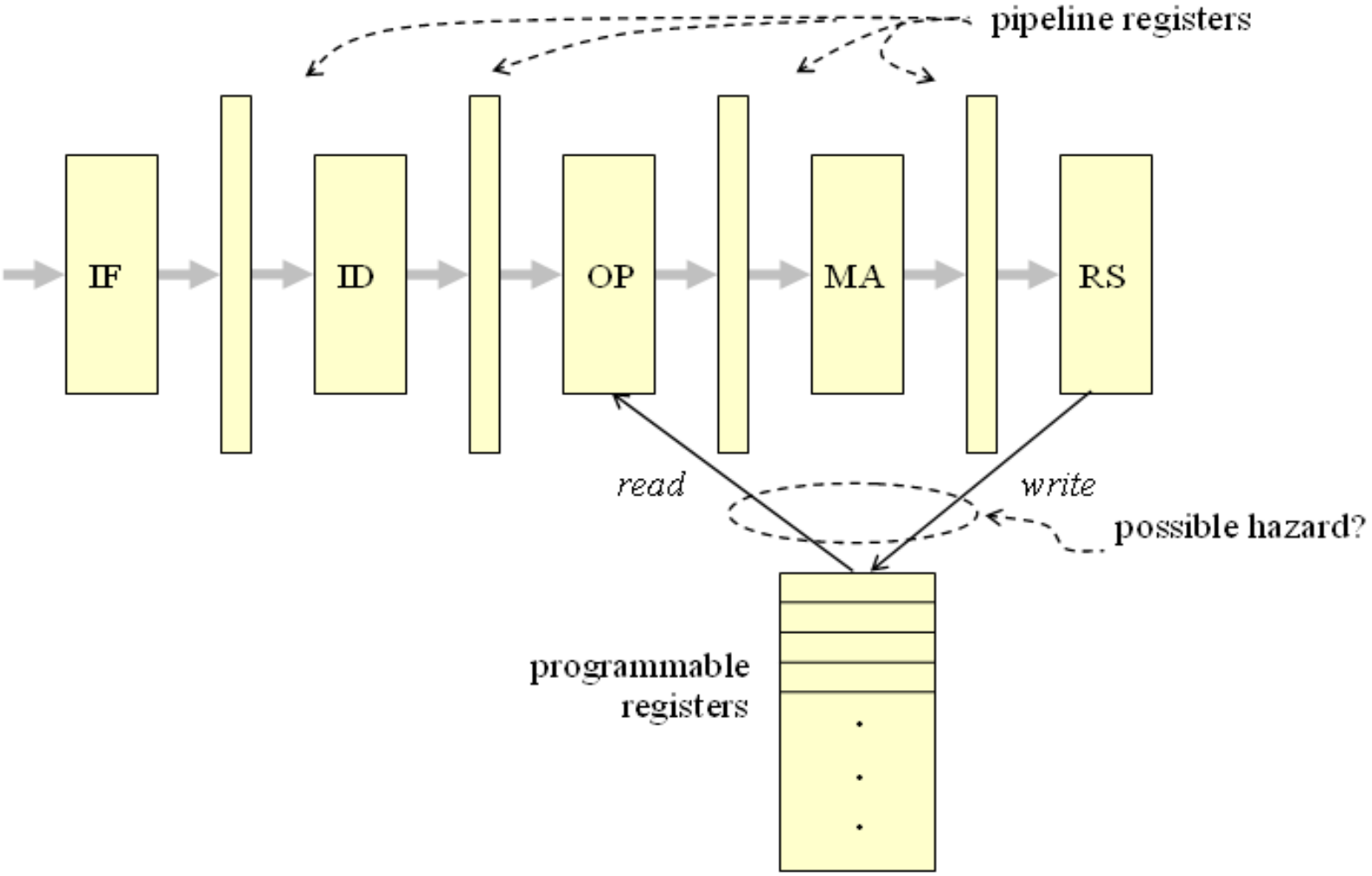
- To achieve this target, designers try to reduce the occurrence and/or performance impact of pipeline *flushes* and *stalls*.

- The design objectives scale up if the processor has been provided with multiple functional units.
  - On such processors, the designers aim to achieve a maximum performance which exceeds one instruction per processor clock cycle.

RISC processors:

- RISC makes possible processor hardware design which achieves higher execution instruction rates.
- The pipelined RISC processor has internal architecture which resembles the five stage pipeline shown earlier; the time per stage is one processor clock cycle.
- *Pipeline registers* are required between stages, so that the relevant output from one stage is passed to the next stage in the pipeline register between the two stages.
- The next figure shows the structure of a pipelined RISC processor.

- Also shown is the bank of programmable registers of the processor. The registers may be accessed by two or more stages of the pipeline.



- A RISC processor must have a *load & store* type of instruction set, which ensures that:

  (i) In execute phase, an instruction makes at most one memory reference for data, and

  (ii) A memory reference instruction does not involve an ALU operation. .

- RISC processors are designed with *hardwired* control unit to achieve the design objective of 'one instruction per clock cycle', or greater.

Summary

- Registers, data paths, and functional units - basic hardware elements which define processor architecture.
- The control unit of the processor; role of control signals.
- Role of stack and stack pointer register.
- Use of cache memory to overcome speed mismatch between processor and main memory.
- Instruction pipeline; instruction latency and throughput; effect of pipeline stalls and flushes. RISC processor.