UNIT III

PIPELINING

- Basic concepts
- Data hazards
- Instruction hazards
- Influence on instruction sets
- Data path and control considerations
- Performance considerations
- Exception handling

Basic Concepts

It is a particularly effective way of organizing concurrent activity in a computer system.

Sequencial execution

Hardware organization

An inter-stage storage buffer, B1, is needed to hold the information being passed from one stage to the next.

New information is loaded into this buffer at the end of each clock cycle. F Fetch: read the instruction from the memory

D Decode: decode the instruction and fetch the source operand(s)

E Execute: perform the operation specified by the instruction

W Write: store the result in the destination location

Pipelined execution

- In the first clock cycle, the fetch unit fetches an instruction I_1 (step F_1) and stores it in buffer B1 at the end of the clock cycle.
- In the second clock cycle the instruction fetch unit proceeds with the fetch operation for instruction I_2 (step F_2).
- Meanwhile, the execution unit performs the operation specified by instruction I_1,which is available to it in buffer B1 (step E_1).
- By the end of the second clock cycle, the execution of instruction I_1 is completed and instruction I_2 is available.
- Instruction I_2 is stored in B1, replacing I_1, which is no longer needed.
- Step E_2 is performed by the execution unit during the third clock cycle, while instruction I_3 is being fetched by the fetch unit.

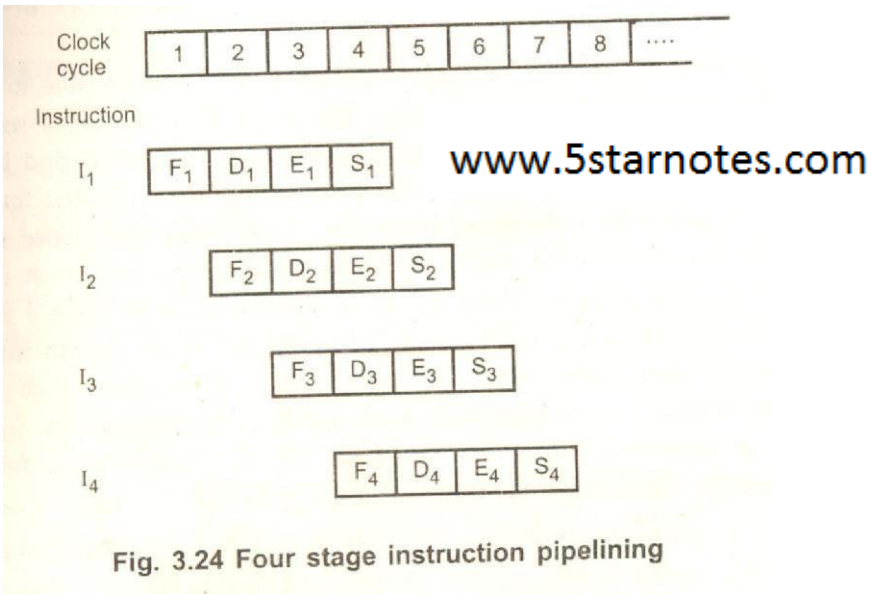


Fig. 3.24 Four stage instruction pipelining

- Four instructions are in progress at any given time.
- So it needs four distinct hardware units.

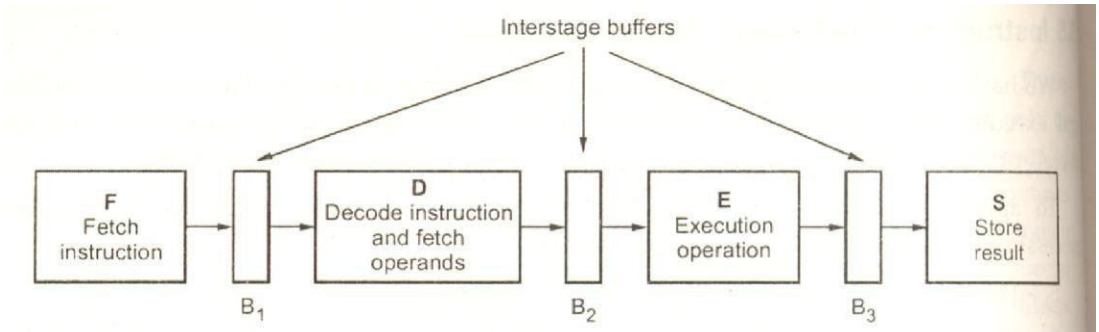


Fig. 3.25 Hardware organisation for four-stage instruction pipeline

3.8.1 Hazards in Instruction Pipelining

These units must be capable of performing their tasks simultaneously and without interfering with one another.

Information is passed from one unit to the next through a storage buffer.
During clock cycle 4, the information in the buffers is as follows:

- Buffer B1 holds instruction I_3, which was fetched in cycle 3 and is being decoded by the instruction-decoding unit.
- Buffer B2 holds both the source operands for instruction I_2 and the specifications of the operation to be performed. This is the information produced by the decoding hardware in cycle 3.
  - The buffer also holds the information needed for the write step of instruction I_2(step W_2).
  - Even though it is not needed by stage E, this information must be passed on to stage W in the following clock cycle to enable that stage to perform the required write operation.
- Buffer B3 holds the results produced by the execution unit and the destination information for instruction I_1.

Pipeline performance

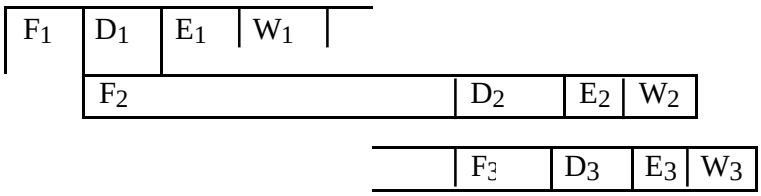
- Pipelining is proportional to the number of pipeline stages.
- For variety of reasons, one of the pipeline stages may not be able to complete its processing task for a given instruction in the time allotted.
- For eg, stage E in the four-stage pipeline is responsible for arithmetic and logic operations and one clock cycle is assigned for this task.
- Although this may be sufficient for most operations some operations such as divide may require more time to complete.
- Instruction I_2 requires 3 cycles to complete from cycle 4 through cycle 6.
- Thus in cycles 5 and 6 the write stage must be told to do nothing, because it has no data to work with.
- Meanwhile, the information in buffer B2 must remain intact until the execute stage has completed its operation.

- This means that stage 2 and in turn stage1 are blocked from accepting new instructions because the information in B1 cannot be overwritten.
- Thus steps D_4 and F_5 must be postponed.

❖ The pipeline may also be stalled because of a delay in the availability of an instruction.

❖ For example, this may be a result of a miss in the cache, requiring the instruction to be fetched from the main memory.

❖ Such hazards are often called control hazards or instruction hazards. Instruction execution steps in successive clock cycles:



Function performed by each process stage in successive clock cycles

Clock Cycle	1	2	3	4	5	6	7	8	9
Stage									
F: Fetch	F_1	F_2	F_2	F_2	F_2				
D: Decode		D_1	idle	idle	idle	D_2	D_3		
E: Execute			E_1	idle	idle	idle	E_2	E_3	
W : Write				W_1	idle	idle	idle	W_2	W_3

- Instruction I_1 is fetched from the cache in cycle$_1$,and its execution proceeds normally.
- Thus in cycles 5 and 6, the write stage must be told to do nothing, because it has no data to work with.
- Meanwhile ,the information in buffer B2 must remain intact until the execute stage has completed its operation.
- This means that stage 2 and in turn stage1 are blocked from accepting now instructions because the information in B1 cannot be overwritten.
- Thus steps D4 and F4 must be postponed.
- Pipelined operation is said to have been stacked for two clock cycles.
- Normal pipelined operation resumes in cycle 7.

Hazard:

Any location that causes the pipeline to stall is called hazard.

Data Hazard

A data hazard is any conditions in which either the source or the destination operands of an instruction are not available at the time expected in the pipeline. As a result some operation has to be delayed ,and the pipeline stalls.

b) Pipeline performance

For a variety of reasons, one of the pipeline stages may not be able to complete its processing task for a given instruction I the time allotted.

For Ex. Stage E in the 4 stage pipeline is responsible for arithmetic and logic operations and one clock cycle is assigned for this task.

Although this may be sufficient for most operations, same operations such as divide may require more time to complete.

Effect of an execution operation talking more than one clock cycle:

- The operation specified in instruction I2 requires three cycles to complete ,from cycle 4 through cycle 6.

Pipe Lining:

1. Basic Concepts:

Pipelining is a particularly effective way of organization concurrent activity in a computer system.

Instruction pipeline:

The fetch, decode and execute cycles for several instructions are performed simultaneously to reduce overall processing time. This process is referred to as instruction pipelining.

Consider 4 stage process

F → Fetch : read the instruction from the memory

D→ Decode: Decode the instruction and fetch the some operands

E→ Execute: perform the operation specified by the instruction.

W→ Write: Store the results in the destination location.

a) Instruction execution divided into 4 steps:

During

Influence on Instruction sets

1. Addressing modes

Many processors provide various combinations of addressing modes such as index,indirect,auto increment, auto decrement and so on.

We can classify these addressing modes as simple addressing modes and complex addressing modes.

A complex address mode may require several accesses to the memory to reach the named operand while simple addressing mode requires only one access to the memory to reach the named operand.

Consider the instruction Load R1,(X(R0)).

This instruction has a complex addressing mode because this instruction requires two memory accesses one to read location X+[R0] and then to read location [X+[R0]].

If R1 is a source operand in the next instruction that instruction has to be stalled for two cycles.

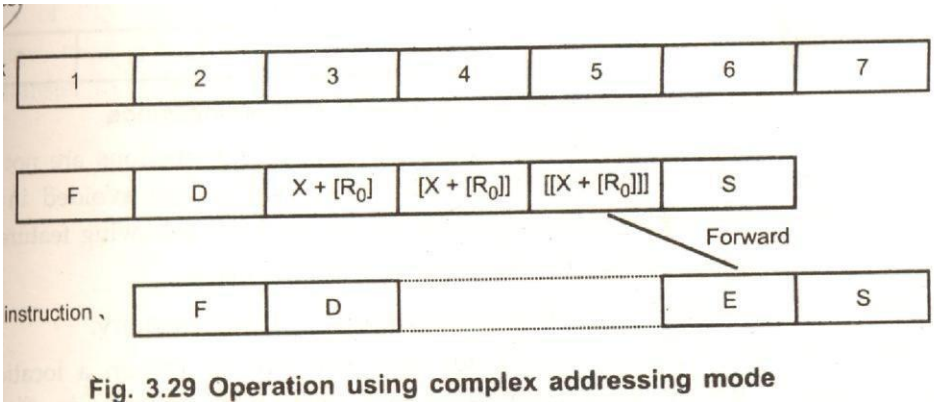


Fig. 3.29 Operation using complex addressing mode

If we want to perform the same operation using simple addressing mode instructions we require to execute three instructions:

ADD R1,R0,#X
LOAD R1,(R2)
LOAD R1,(R1)

The ADD instruction performs the operation R1 ← [R0]+X.

The two load instructions fetch the address and then the operand from the memory.

The two load instructions fetch the address and then the operand from the memory.

This sequence of instructions takes exactly the same number of clock cycles as the single load instruction having a complex addressing mode.

The above example indicates that in a pipeline processor, complex addressing mode do not necessarily lead to faster execution.

But it reduce the number of instructions needed to perform particular task and hence reduce the program space needed in the main memory.

Disadvantages

a) Long execution times of complex addressing mode instructions may cause pipeline to stall.

b) They require more complex hardware to decode and execute them.

c) They are not convenient for compilers to work with.

Due to above reasons the complex addressing mode instructions are not suitable for pipelined execution.

These addressing modes are avoided in modern processors.

Features of modern processor addressing modes

a) They access operand from the memory in only one cycle.

b) Only load and store instructions are provided to access memory.

c) The addressing modes used do not have side effects.(when a location other than one explicitly named in an instruction as a destination operand is affected, the instruction is said to have a side effect).

2. Condition codes

Most of the processors store condition code flags in their status register. The conditional branch instruction checks the condition code flag and the flow of program execution is decided.

When the data dependency or branch exists, the pipeline may stall.

A special optimizing compiler is designed which reorders the instructions in the program which is to be executed.

This avoids the stalling the pipeline, the compiler takes the care to obtain the correct output while reordering the instructions.

To handle condition codes,

a) The condition code flags should be affected by as few instructions as possible. It provides flexibility in reordering instructions.

Otherwise, the dependency produced by the condition code flags reduces the flexibility available for the compiler reorder instructions.

b) It should be known to compiler that in which instructions of a program the condition codes are affected and in which they remain unaffected.

Datapath and control considerations

When single bus is used in a processor only one data word can be transferred over the bus in a clock cycle.

This increases the steps required to complete the execution of the instruction. To reduce the number of steps needed to execute instructions most commercial processors provide multiple internal paths that enable several transfer to take place in parallel.

Modified 3 bus structure of the processor for pipelined execution

3 buses are used to connect registers and the ALU of the processor.
All general purpose registers are connected by a single block called register file.

It has 3 ports:

- One input port
- Two output ports

It is possible to access data of 3 register in one clock cycle, the value can be loaded in one register from bus C and data from two register can be accessed to bus A and bus B respectively.

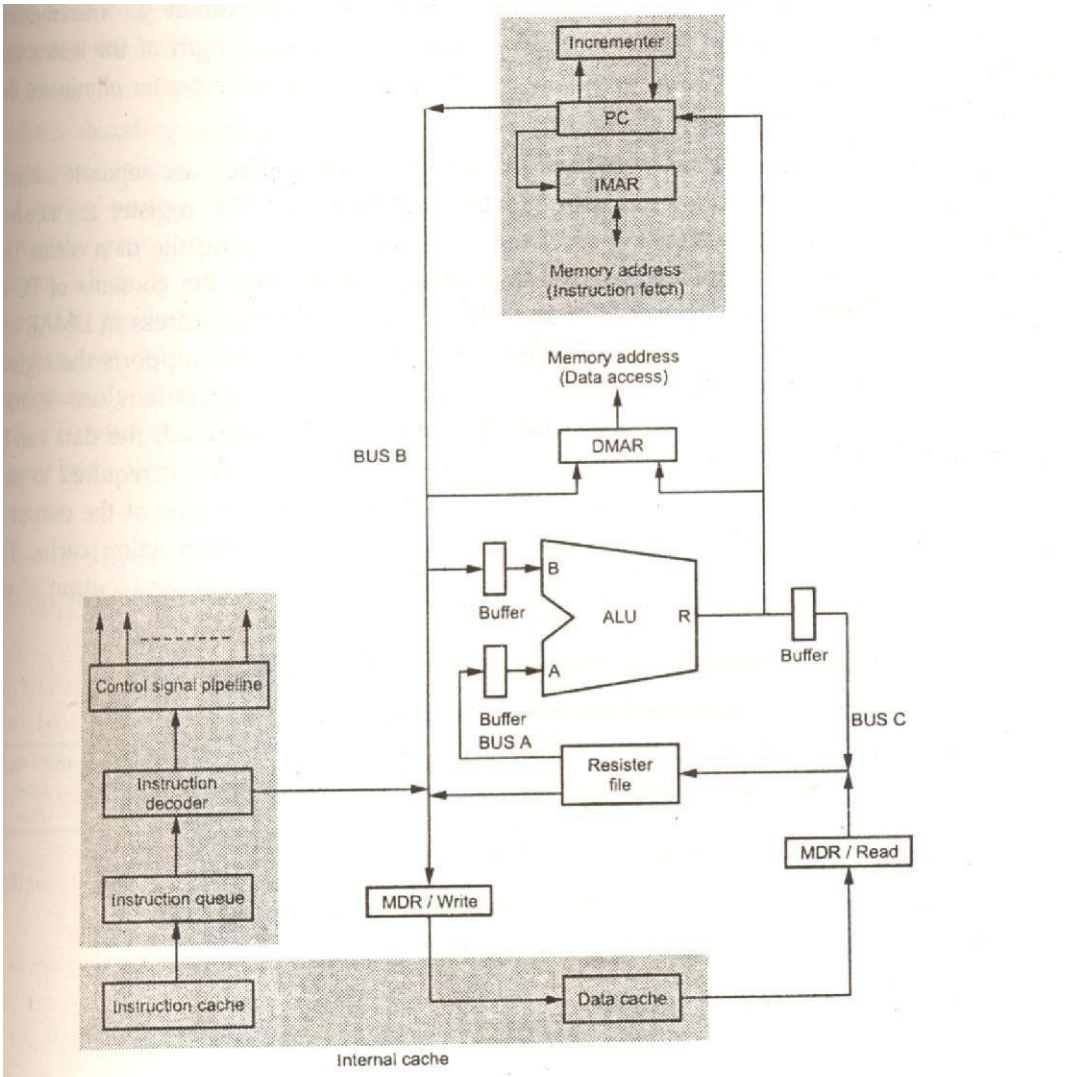


Fig. 3.31 Modified three-bus structure of the processor for pipelined execution

Buses A and B are used to transfer the source operands to the A and B inputs of the ALU.

After performing arithmetic or logic operation result is transferred to the destination operand over bus C.

To increment the contents of PC after execution of each instruction to fetch the next instruction separate unit is provided, it is known as incrementer.

Incrementer increments the contents of PC accordingly to the length of the instruction so that it can point to next instruction in the sequence.

The processor uses separate instruction and data caches.

They use separate address and data connections to the processor.

Two versions of MAR register are needed IMAR for accessing the instruction cache and DMAR for accessing the data cache.

The PC is connected directly to the IMAR.

This allows transferring the contents of PC to IMAR and performing ALU operation simultaneously.

The data address in DMAR can be obtained directly from the register file or from the ALU.

It supports the register indirect and indexed addressing modes.

The read and write operations use separate MDR registers.

When load and store operations are to be performed the data can be transferred directly between these registers file.

It is not required to pass this data through the ALU.

Two buffer registers at the input and one at the output of the ALU are used. The instruction queue gets loaded from instruction cache.

The output of the queue is connected to the instruction decoder input and output of the decoder is connected to the control signal pipeline.

The processor can perform the following operations independently,

1. Reading an instruction from the instruction cache.
2. Incrementing the PC.
3. Decoding an instruction by instruction decoder.
4. Reading from and writing into the data cache.
5. Reading the contents of up to two registers from the register file.
6. Writing into one register in the register file.
7. Performing an ALU operation.

Superscalar Operation

- Several instructions can be executed concurrently because of pipelining. However, these instructions in the pipeline are in different stages of execution such as fetch, decode, ALU operation.
- When one instruction is fetch phase, one instruction is completing its execution in the absence of hazards.
- Thus at the most, one instruction is executed in each clock cycle.
- This means the maximum throughput of the processor is one instruction per clock cycle when pipelining is used.
- Processors reach performance levels greater than one instruction per cycle by fetching, decoding and executing several instructions concurrently.

- This mode of operation is called superscalar.
- A processor capable of parallel instruction per cycle is known as superscalar processor.
- A superscalar processor has multiple execution units (E-units), each of which is usually pipelined, so that they constitute a set of independent instruction pipelines.
- Its program control unit (PCU) is capable of fetching and decoding several instructions concurrently.
- It can issue multiple instructions simultaneously.
- If the processor can issue up to k instructions simultaneously to the various E-units, then k is called instruction issue degree.
- It can be six or more using current technology.

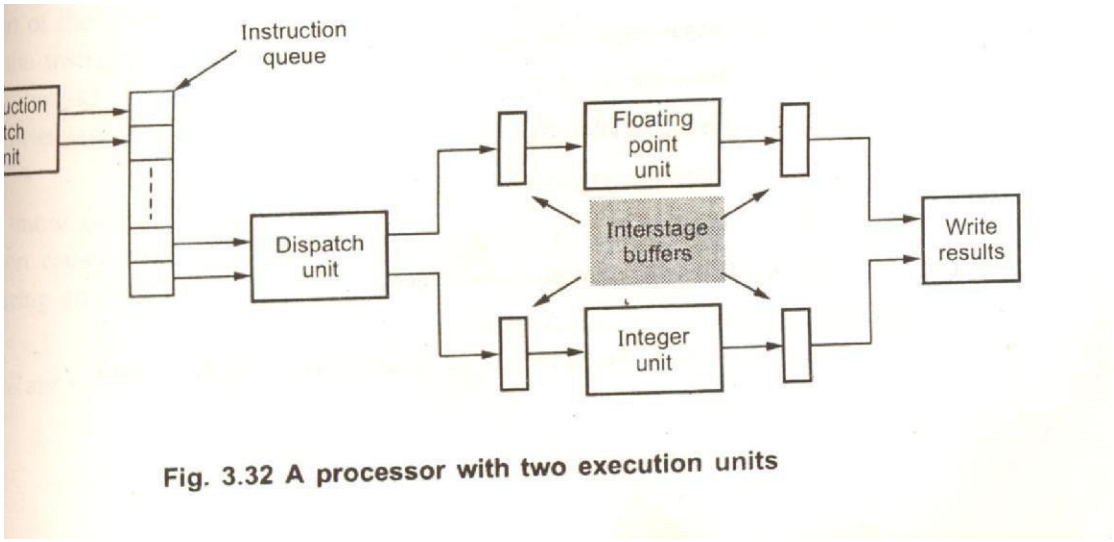
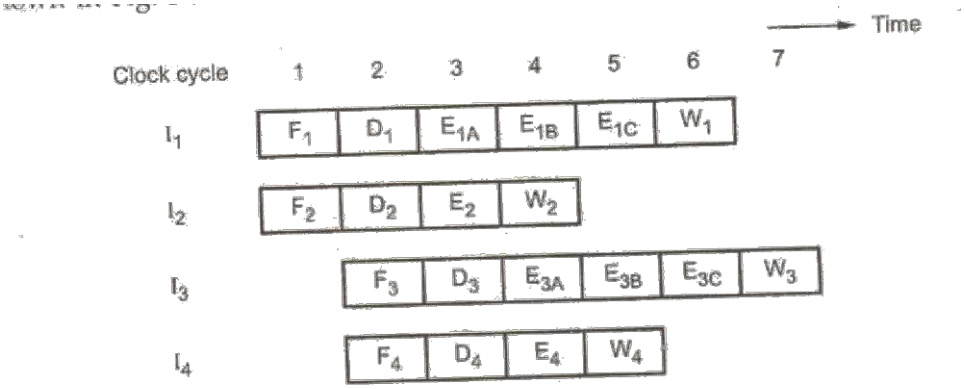


Fig. 3.32 A processor with two execution units

- A processor has two execution units,
  - i) Integer unit (used for integer operations)
  - ii) Floating point unit (used for floating point operations)

- The instruction fetch unit can read two instructions simultaneously and stores them temporarily in the instruction queue.
- The queue operates on the principle first in first out (FIFO).
- In each clock cycle, the dispatch unit can take two instructions from the instruction queue and decodes them.
- If these instructions are such that one is integer and another is floating-point with no hazards, then both instructions are dispatched in the same clock cycle.
- The compiler is designed to avoid hazards by proper selection and ordering of the instructions.



Note : I_1, I_3 : Floating point Instruction

I_2 : I_4 : Integer instructions

F : Instruction fetching

D : Instruction decoding

E : Instruction execution

W : Write results

E_A, E_B, E_C : Instruction execution stages

It is assumed that no hazards are encountered.

Fig. 3.33. An example of instruction execution flow

- It is

takes one clock cycle to complete execution.

1. The instruction fetch unit fetches instructions I1 and I2 in clock cycle.
2. I1 and I2 enter the dispatch unit in cycle 2. The fetch unit fetches next two instructions, I3 and I4 during the same cycle.
3. The floating point unit takes three clock cycles to complete the floating point operation(Execution: EA,EB,EC) specified in I1. The integer unit completes execution of I2 in one clock cycle(clock cycle 3). Also, instructions I3 and I4 enter the dispatch unit in cycle 3.
4. We have assumed the floating point unit as a three-stage pipeline. So it can accept a new instruction for execution in each clock cycle. So during clock cycle 4, though the execution of I1 is in progress, it accepts I3 for the execution. The integer unit can accept a new instruction for execution because instruction I2 has entered to the write stage. Thus, instructions I3 and I4 are dispatched in cycle 4. Instruction I1 is still in the execution phase, but in the second stage of the internal pipeline in floating-point unit.
5. The instructions complete execution with no hazards as shown in further clock cycles.

Out-of-order Execution

- The dispatch unit dispatches the instructions in the order in which they appear in the program. But their execution may be completed in the different order.
- For example, execution of I2 is completed before the complete execution of I1.
- Thus the execution of the instructions is completed out of order.
- If there is a data dependency among the instructions, the execution of the instructions is delayed.
- For example, if the execution of I2 needs the results of execution of I1, I2 will be delayed.
- If such dependencies are handled correctly, the execution of the instructions will not be delayed.
- There are two causes of exceptions,
  - a bus error (during an operand fetch)
  - illegal operation(eg. Divide by zero)
  -

2 types of exceptions,
- Imprecise exceptions
- Precise exceptions

Imprecise exception

Consider the pipeline timing, the result of operation of I2 is written into the register file in cycle 4.

If instruction I1 causes an exception and succeeding instructions are permitted to complete execution, then the processor is said to have imprecise exceptions. Because of the exception by I1, program execution is in an inconsistent state.

Precise exception

In the imprecise exception, consistent state is not guaranteed when an exception occurs.

If the exception occurred then to guarantee a consistant state, the result of the execution of instructions must be written into the destination locations strictly in the program order. The result of execution of I2 is written to the destination in cycle 4(W2).

But the result of I1 is written to the destination in cycle
6(W1). So step W2 is to be delayed until cycle 6.

The integer execution unit has to retain the result of execution of I2.
So it cannot accept instruction I4 until cycle 6.

Thus in precise exception, if the exception occurs during an instruction, the succeeding instructions, may have been partially executed are discarded.

If an external interrupt is received, the dispatch unit stops reading new instructions from the instruction queue.
The instructions which are placed in the instruction queue are discarded.

The processor first completes the pending execution of the instructions to completion.
The consistent state of the processor and all its registers is achieved.

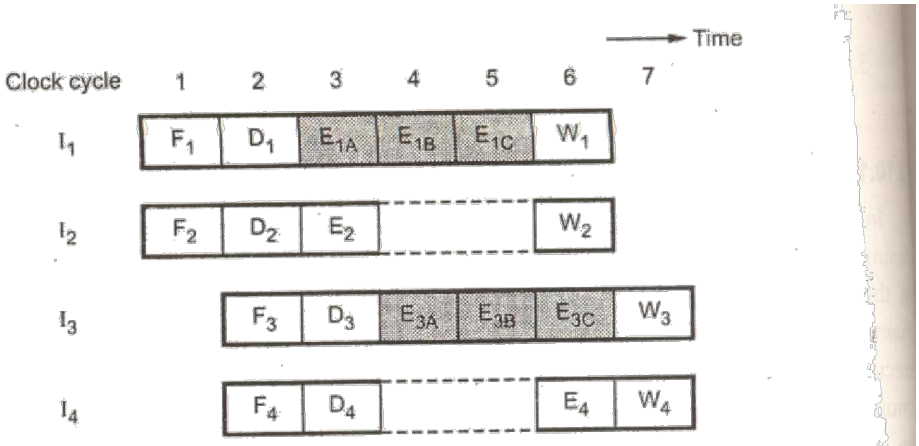


Fig. 3.34 Instruction execution completion in program order with delayed write

Then the processor starts the interrupt handling process.

Execution completion

The decoding instructions are stored temporarily into the temporary registers and later they are transferred to the permanent registers in correct program order. Thus 2 write operations TW and W respectively are carried out.

The step W is called the commitment step because the final result gets stored into the permanent register during this step.

Eventhough, exception occurs, the result of succeeding instructions would be in temporary registers and hence can be safely discarded.

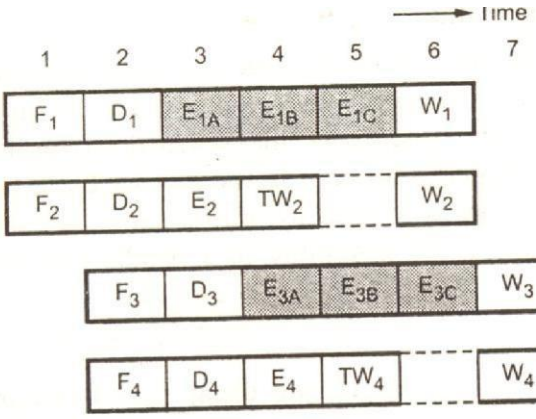


Fig. 3.35 Instruction execution completion in program order using temporary registers

lote : TW : Write into a temporary register

W : Transfer contents of TW into permanent register

Register renaming

- When temporary register holds the contents of the permanent register, the name of permanent register is given to that temporary register is called as register renaming.
- For example, if I2 uses R4 as a destination register, then the temporary register used in step TW2 is also reffered as R4 during cycles 6 and 7, that temporary register used only for instructions that follow I2 in program order.
- For example, if I1 needs to read R4 in cycle 6 or 7, it has to access R4 though it contains unmodified data be I2.

Commitment Unit

It is a special control unit needed to guarantee in-order commitment when out-of-order execution is allowed.

It uses a special queue called the reorder buffer which stores the instruction committed next.

Retired instructions

- The instructions entered in the reorder buffer(queue)of the commitment unit strictly in program order.
- When the instruction from this queue is executed completely, the results obtained from it are copied from temporary registers to the permanent registers and instruction is removed from the queue.
- The resources which were assigned to the instruction are released.
- At this stage, the instruction is said to have been retired.
- The instructions are retired in program order though they may complete execution out of order.
- That is, all instructions that were dispatched before it must also have been retired.

Dispatch operation

Each instruction requires the resources for its execution.

The dispatch unit first checks the availability of the required resources and then only dispatches the instructions for execution.

These resources include temporary register, a location in the order buffer, appropriate execution unit etc.

Deadlock

Consider 2 units U1 and U2 are using shared resources.

U2 needs completion of the task assign to unit U1 to complete its task.

If unit U2 is using a resource which is also required to unit U1, both units cannot complete the tasks assigned to them.

Both the units remain waiting for the need resource.

Also, unit U2 is waiting for the completion of task by unit U1 before it can release that resource. Such a situation is called a deadlock.