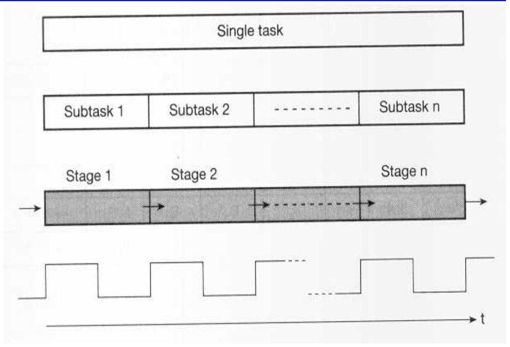
5 Pipelined Processor

✈temporal overlapping of processing, assembly line

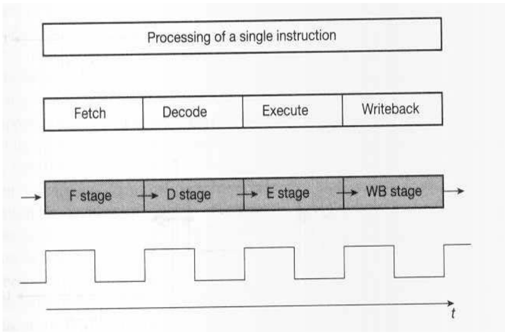
- 5.1 Basic concept
- 5.2 Design space of pipelines
- 5.3 Overview of pipelined instruction processing
- 5.4 Pipelined execution of integer and Boolean instructions
- 5.5 Pipelined processing of loads and stores

TECH
Computer Science

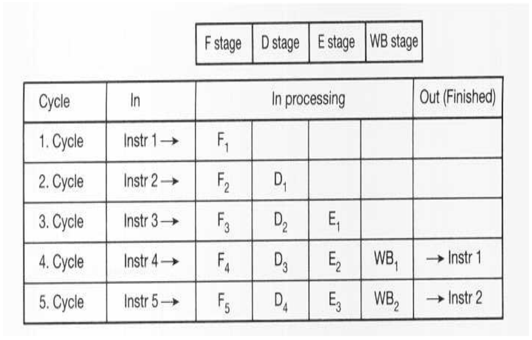
5.1.1 Principle of pipelining



Principle of pipelining e.g.

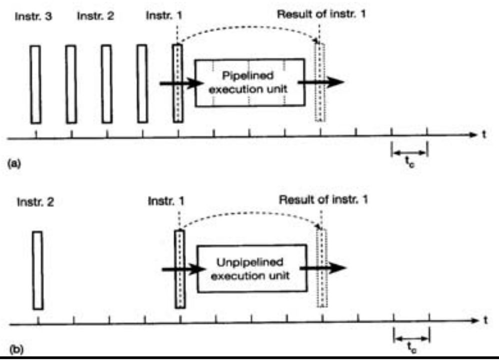


Processing of a sequence of instructions using a basic pipeline

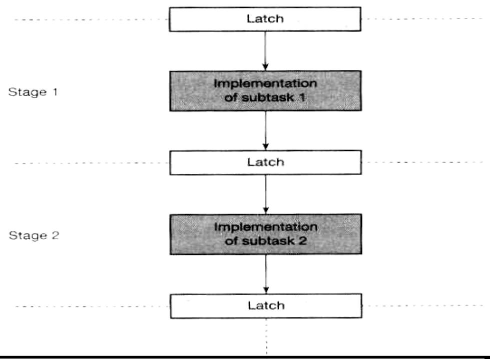

F stage
D stage
E stage
WB stage

Cycle	In	In processing				Out (Finished)
1. Cycle	Instr 1 →	F_1				
2. Cycle	Instr 2 →	F_2	D_1			
3. Cycle	Instr 3 →	F_3	D_2	E_1		
4. Cycle	Instr 4 →	F_4	D_3	E_2	WB_1	→ Instr 1
5. Cycle	Instr 5 →	F_5	D_4	E_3	WB_2	→ Instr 2

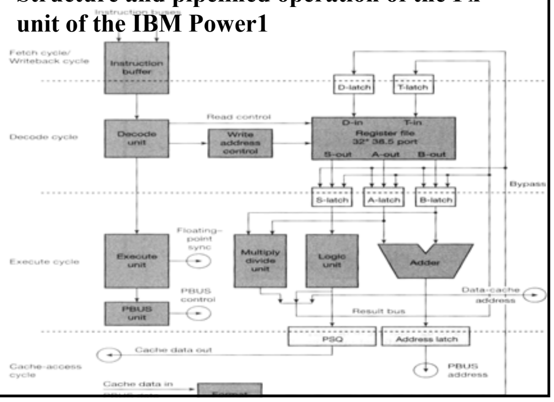
Pipelined and unpipelined processing



5.1.2 General structure of pipelines



Structure and pipelined operation of the Fx unit of the IBM Power1



Pipeline Performance Measures

- Cycle time: t_c
  - ✈ **is determined by the worst-case processing time of the longest stage**
- Repetition Rate: R
  - ✈ **the shortest possible time interval between subsequent independent instructions in the pipeline**
- Performance potential of a pipeline: P

  $P = 1/(R * t_c)$
- PowerPC603 FP double Mul. e.g. $R = 2$, $t_c = 12$ nsec

  $P = 1/(R * t_c) = 1/(2*12nec) = 44.6$ MFLOPS

Performance: RAW-dependent

- Latency:
  - ✈ **specifies the amount of time that the result of a particular instruction takes to become available in the pipeline for a subsequent dependent instruction.**
- Define-use latency (10 to 100 cycles)
  - ✈ **mul r1, r2, r3**
  - ✈ **add r5, r1, r4**
- Load-use latency (1 to 3 cycles)
  - ✈ **load r1, x**
  - ✈ **add r5, r1, r2**
- Stalled: the immediately following RAW-dependent instruction has to be **stalled** in the pipeline for n-1 cycle

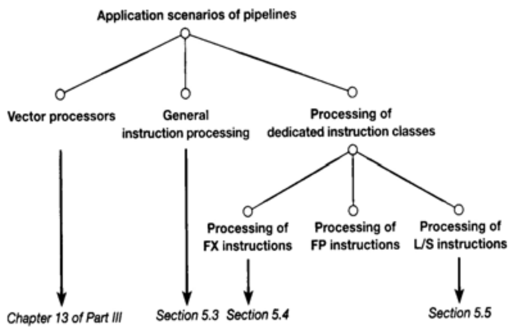
Improve Performance

- **Multiple-operation instructions**
- HP PA 7100

  FMPYADD RM1, RM2, RM3, RA1, RA2

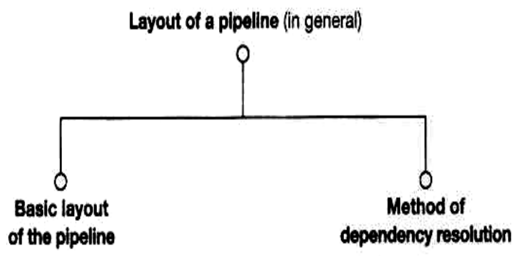
  RM3←RM1*RM2   RA2←RA1+RA2
- PowerPC
  - ✈ **FMA for performing (A*C) + B**

5.1.4 Application scenarios of pipelines


Application scenarios of pipelines
Vector processors
General instruction processing
Processing of dedicated instruction classes
Processing of FX instructions
Processing of FP instructions
Processing of L/S instructions
Chapter 13 of Part III
Section 5.3
Section 5.4
Section 5.5

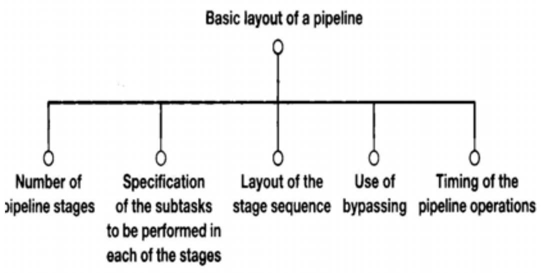
5.2 Design space of pipelines

- key aspect of the design space of pipeline

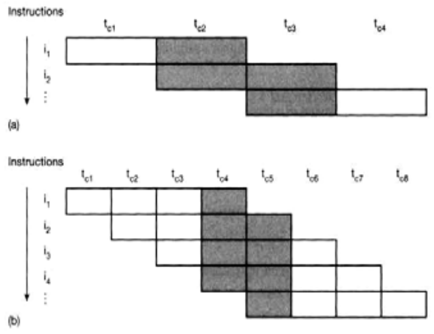

Layout of a pipeline (in general)
Basic layout of the pipeline
Method of dependency resolution

5.2.2 Basic layout of a pipeline

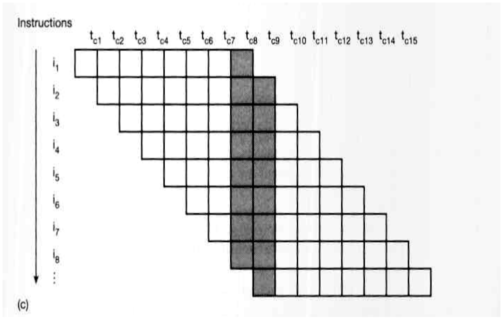
- Design space of the overall stage layout


Basic layout of a pipeline
Number of pipeline stages
Specification of the subtasks to be performed in each of the stages
Layout of the stage sequence
Use of bypassing
Timing of the pipeline operations

Increasing parellelism by raising the number of pipeline stages



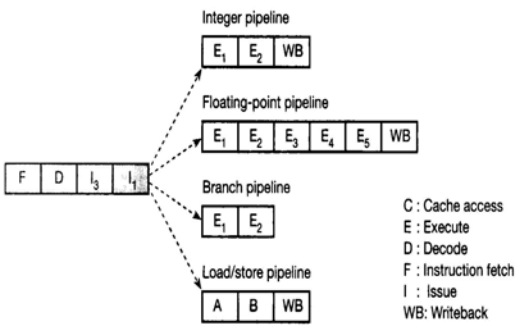
Eight -stage pipeline



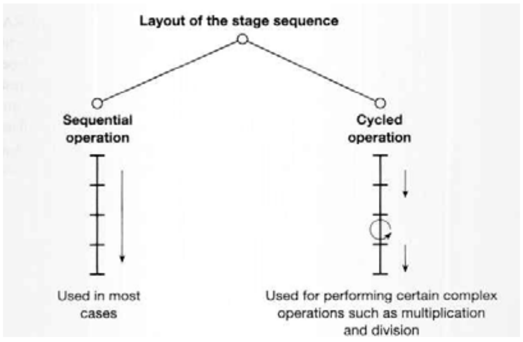
Problems arise for more stages

- data and control dependencies occur more frequently
  - **stalled and wait for data**
  - **reload pipe in case of branch**
- subtask becomes less balances (in execution time)
  - **cycle time is determined by the worst-case processing time of the longest stage**
- In most case
  - **5-10 stages**

Pipelines e.g. DEC α 21064



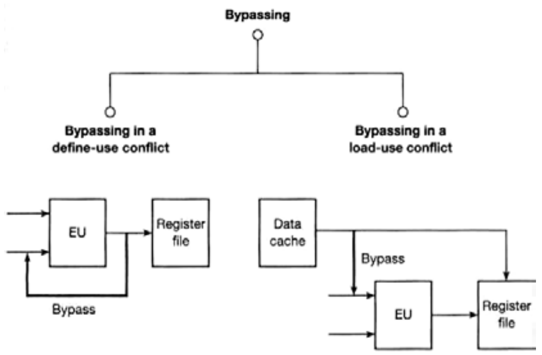
Layout of the stage sequence


Layout of the stage sequence
Sequential operation
Cycled operation
Used in most cases
Used for performing certain complex operations such as multiplication and division

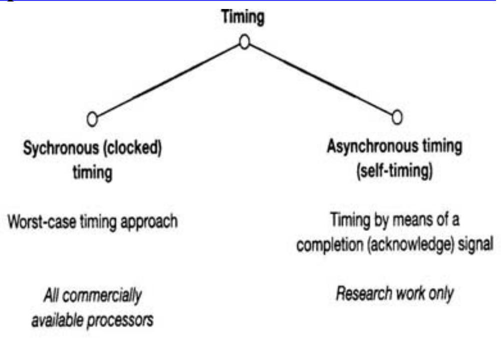
Bypasses (data forwarding in RAW)

- Unless special arrangements are made,
- the results of the operation instruction is written into the register file, or into the memory,
- and then it is fetched from there as a source operand.

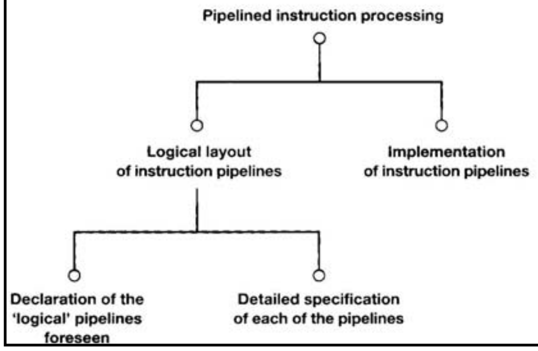
Principle of bypassing in define-use and load-use conflicts


Bypassing
Bypassing in a define-use conflict
Bypassing in a load-use conflict
EU
Register file
Bypass
Data cache
Bypass
EU
Register file

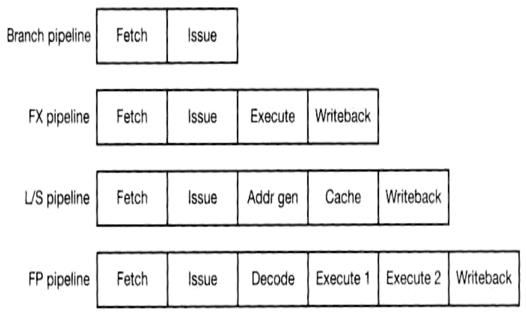
Possibilities for the timing of pipeline operation


Timing
Sychronous (clocked) timing
Asynchronous timing (self-timing)
Worst-case timing approach
Timing by means of a completion (acknowledge) signal
All commercially available processors
Research work only

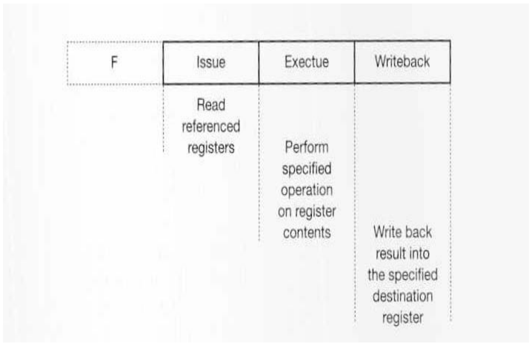
5.3 Overview: pipelined instruction processing


Pipelined instruction processing
Logical layout of instruction pipelines
Implementation of instruction pipelines
Declaration of the 'logical' pipelines foreseen
Detailed specification of each of the pipelines

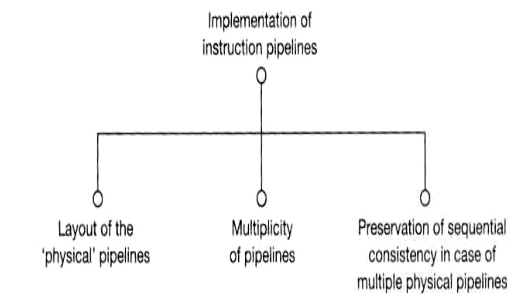
Declaration of Logical Pipeline: e.g. Powerpc 601



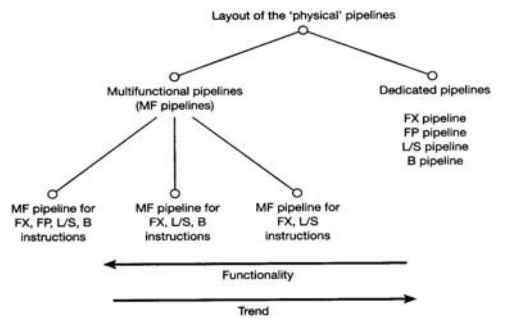
Detailed Specification of each of the pipeline: e.g. //


F
Issue
Exectue
Writeback
Read referenced registers
Perform specified operation on register contents
Write back result into the specified destination register

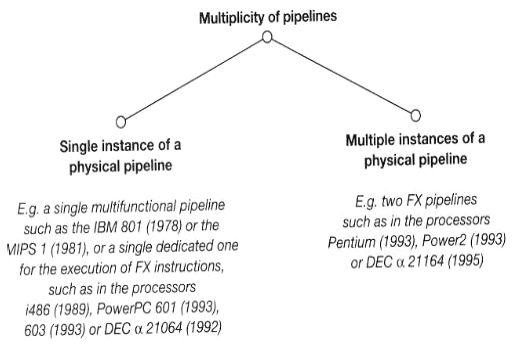
Implementation of instruction pipelines (v.s. logical)


Implementation of instruction pipelines
Layout of the 'physical' pipelines
Multiplicity of pipelines
Preservation of sequential consistency in case of multiple physical pipelines

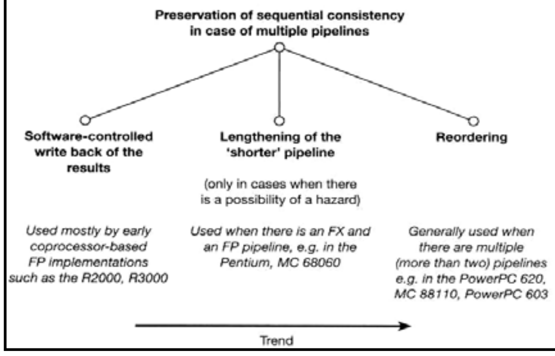
Layout of physical pipelines


Layout of the 'physical' pipelines
Multifunctional pipelines (MF pipelines)
Dedicated pipelines
FX pipeline
FP pipeline
L/S pipeline
B pipeline
MF pipeline for FX, FP, L/S, B instructions
MF pipeline for FX, L/S, B instructions
MF pipeline for FX, L/S instructions
Functionality
Trend

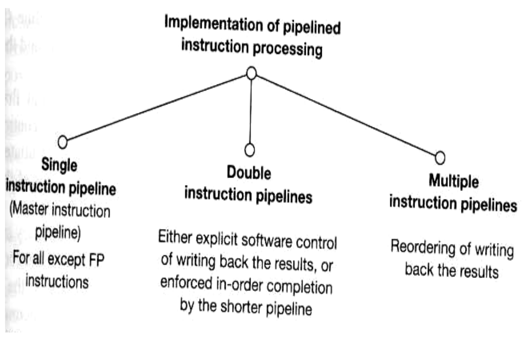
Multiplicity of pipelines


Multiplicity of pipelines
Single instance of a physical pipeline
Multiple instances of a physical pipeline
E.g. a single multifunctional pipeline such as the IBM 801 (1978) or the MIPS 1 (1981), or a single dedicated one for the execution of FX instructions, such as in the processors i486 (1989), PowerPC 601 (1993), 603 (1993) or DEC α 21064 (1992)
E.g. two FX pipelines such as in the processors Pentium (1993), Power2 (1993) or DEC α 21164 (1995)

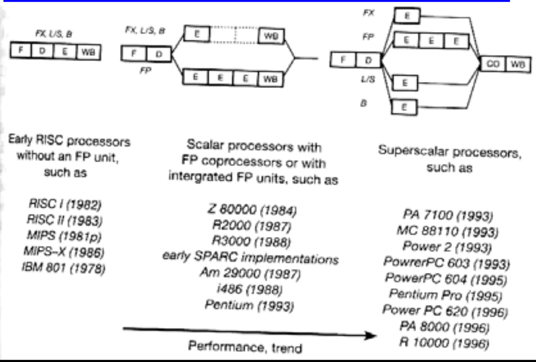
Preserveing sequential consistency


Preservation of sequential consistency in case of multiple pipelines
Software-controlled write back of the results
Lengthening of the 'shorter' pipeline
Reordering
(only in cases when there is a possibility of a hazard)
Used mostly by early coprocessor-based FP implementations such as the R2000, R3000
Used when there is an FX and an FP pipeline, e.g. in the Pentium, MC 68060
Generally used when there are multiple (more than two) pipelines e.g. in the PowerPC 620, MC 88110, PowerPC 603
Trend

Preserveing sequential consistency, implementation e.g.

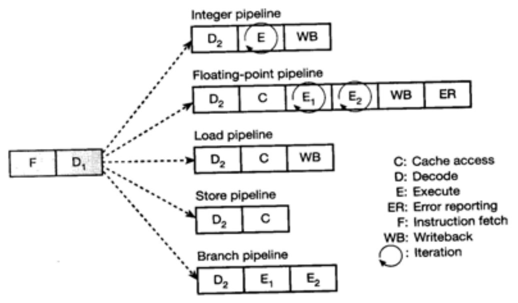

Implementation of pipelined instruction processing
Single instruction pipeline
(Master instruction pipeline)
For all except FP instructions
Double instruction pipelines
Either explicit software control of writing back the results, or enforced in-order completion by the shorter pipeline
Multiple instruction pipelines
Reordering of writing back the results

Preserveing sequential consistency, e.g.

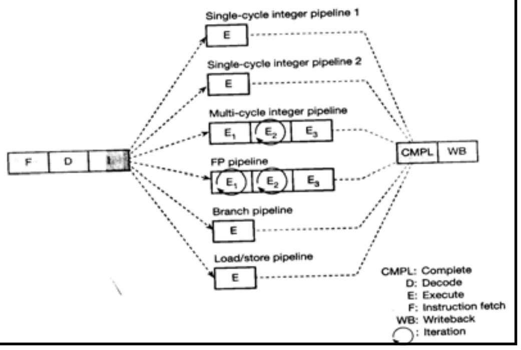


Case studies: Pentium

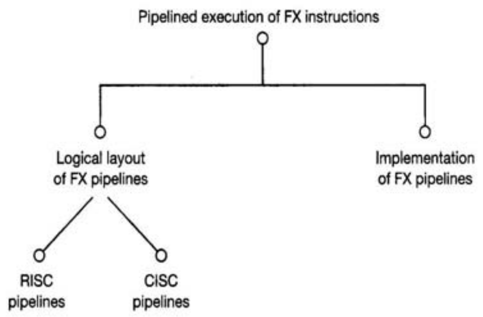
- Logic layout of Pentium’s pipelines



Case studies: PowerPC 604

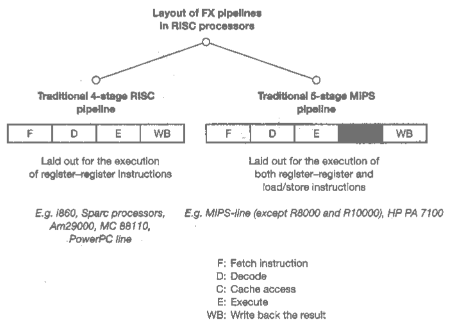


5.4 (Specific) Pipelines execution: Integer and Boolean instructions (FX)



Pipelined execution of FX instructions
Logical layout of FX pipelines
Implementation of FX pipelines
RISC pipelines
CISC pipelines

RISC pipelines 4 or 5 stages



Layout of FX pipelines in RISC processors
Traditional 4-stage RISC pipeline
Traditional 5-stage MIPS pipeline
F D E WB
F D E WB
Laid out for the execution of register–register instructions
Laid out for the execution of both register–register and load/store instructions
E.g. i860, Sparc processors, Am29000, MC 88110, PowerPC line
E.g. MIPS-line (except R8000 and R10000), HP PA 7100
F: Fetch instruction
D: Decode
C: Cache access
E: Execute
WB: Write back the result

Tradictrional FX pipeline of RISC processors

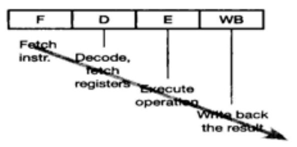


Figure 5.36 Traditional FX pipeline of RISC processors.

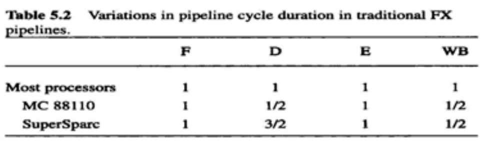
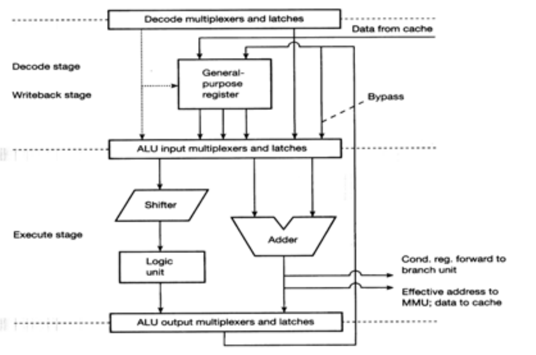


Table 5.2 Variations in pipeline cycle duration in traditional FX pipelines.

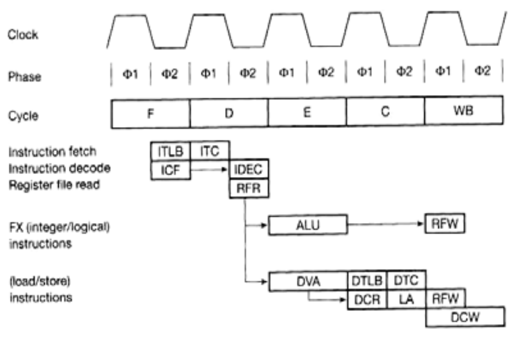
	F	D	E	WB
Most processors	1	1	1	1
MC 88110	1	1/2	1	1/2
SuperSparc	1	3/2	1	1/2

Logical to Physical: e.g. PowerPC601 using a single universal FX unit

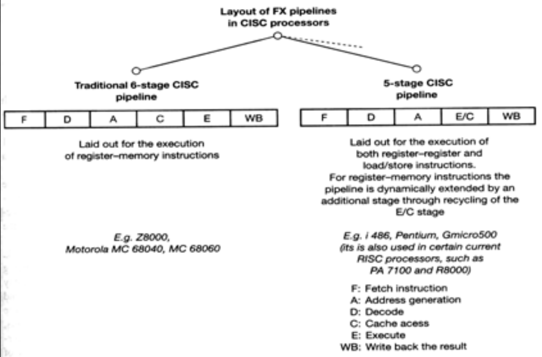


Decode multiplexers and latches
Data from cache
Decode stage
Writeback stage
General-purpose register
Bypass
ALU input multiplexers and latches
Shifter
Execute stage
Adder
Logic unit
Cond. reg. forward to branch unit
Effective address to MMU; data to cache
ALU output multiplexers and latches

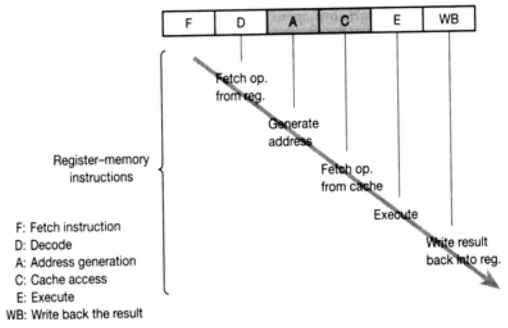
Layout 5 stages e.g. :
FX and L/S pipelines in the MIPS R4200



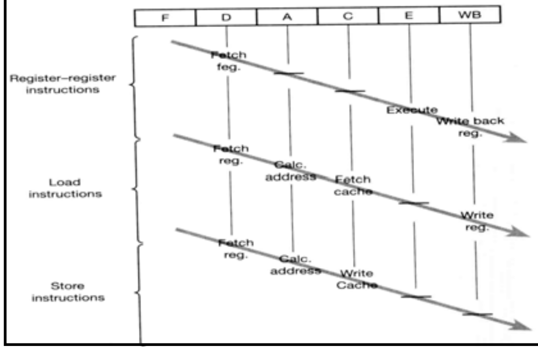
CISC pipeline 6 or 5 stages


Layout of FX pipelines in CISC processors
Traditional 6-stage CISC pipeline
F
D
A
C
E
WB
Laid out for the execution of register–memory instructions
E.g. Z8000, Motorola MC 68040, MC 68060
5-stage CISC pipeline
F
D
A
E/C
WB
Laid out for the execution of both register–register and load/store instructions. For register–memory instructions the pipeline is dynamically extended by an additional stage through recycling of the E/C stage
E.g. i 486, Pentium, Gmicro500 (its is also used in certain current RISC processors, such as PA 7100 and R8000)
F: Fetch instruction
A: Address generation
D: Decode
C: Cache acess
E: Execute
WB: Write back the result

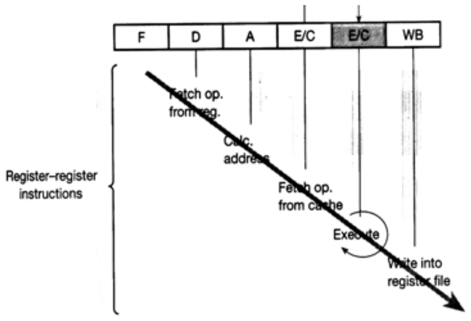
Traditional CISC pipeline:
The execution of reister-memory instruction



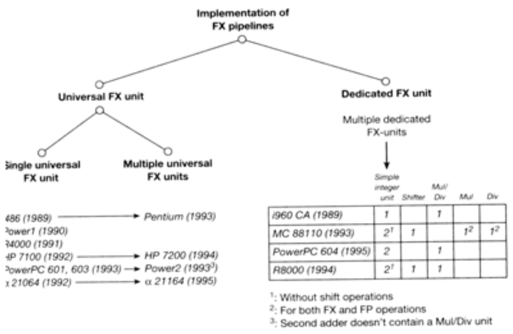
CISC pipeline:
Execution of register-register and load/store instructions



CISC pipeline 5 stage: recycling E/C stage



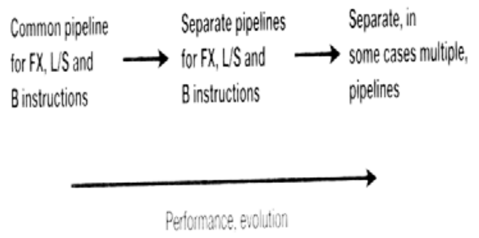
Implementation of FX units: how many



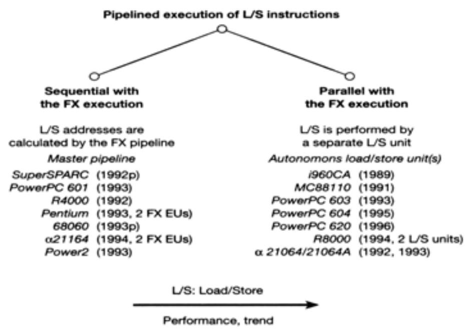
	Simple integer unit	Shifter	Mul/Div	Mul	Div
i960 CA (1989)	1		1		
MC 88110 (1993)	2^1	1		1^2	1^2
PowerPC 604 (1995)	2		1		
R8000 (1994)	2^1	1	1		

¹: Without shift operations
²: For both FX and FP operations
³: Second adder doesn't contain a Mul/Div unit

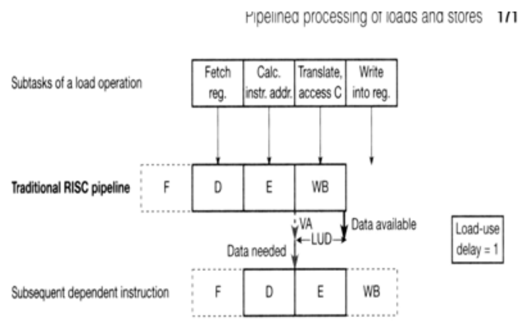
Trend in increasing the performance


Common pipeline for FX, L/S and B instructions
Separate pipelines for FX, L/S and B instructions
Separate, in some cases multiple, pipelines
Performance, evolution

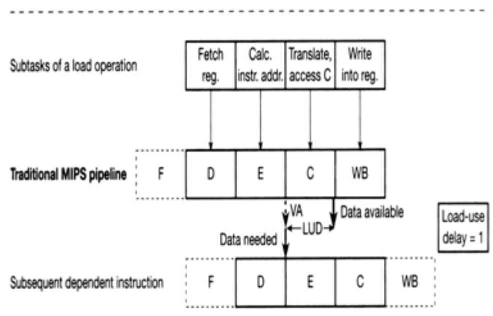
5.5 (Specific) Pipelines execution: loads and stores


Pipelined execution of L/S instructions
Sequential with the FX execution
L/S addresses are calculated by the FX pipeline
Master pipeline
SuperSPARC (1992p)
PowerPC 601 (1993)
R4000 (1992)
Pentium (1993, 2 FX EUs)
68060 (1993p)
α21164 (1994, 2 FX EUs)
Power2 (1993)
Parallel with the FX execution
L/S is performed by a separate L/S unit
Autonomons load/store unit(s)
i960CA (1989)
MC88110 (1991)
PowerPC 603 (1993)
PowerPC 604 (1995)
PowerPC 620 (1996)
R8000 (1994, 2 L/S units)
α 21064/21064A (1992, 1993)
L/S: Load/Store
Performance, trend

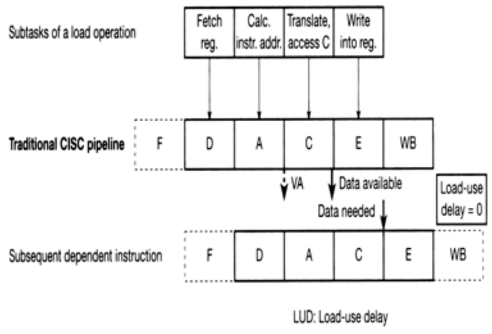
5.5.3 Load-use delay: RICS pipelines


Pipelined processing of loads and stores 171
Subtasks of a load operation
Fetch reg.
Calc. instr. addr.
Translate, access C
Write into reg.
Traditional RISC pipeline
F
D
E
WB
VA
LUD
Data available
Data needed
Load-use delay = 1
Subsequent dependent instruction
F
D
E
WB

Load-use delay: MIPS

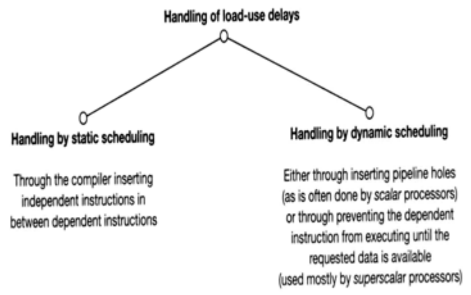


Load-use delay: CISC

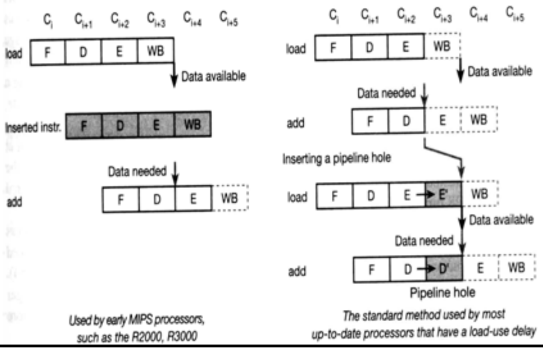


Handling Load-use delay

- Basic approaches to cope with a load-use delay


Handling of load-use delays
Handling by static scheduling
Through the compiler inserting independent instructions in between dependent instructions
Handling by dynamic scheduling
Either through inserting pipeline holes (as is often done by scalar processors) or through preventing the dependent instruction from executing until the requested data is available (used mostly by superscalar processors)

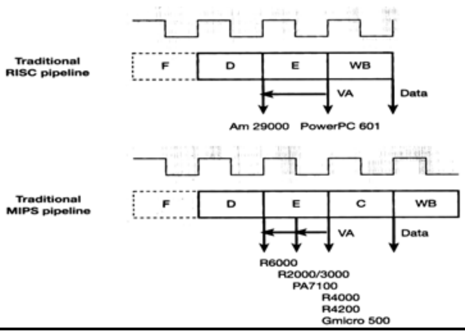
Remove Load-use delay



Used by early MIPS processors, such as the R2000, R3000

The standard method used by most up-to-date processors that have a load-use delay

Remove Load-use delay: bringing forward the claculation of virtual address: for slow cache



Traditional RISC pipeline
F
D
E
WB
VA
Data
Am 29000
PowerPC 601
Traditional MIPS pipeline
C
R6000
R2000/3000
PA7100
R4000
R4200
Gmicro 500