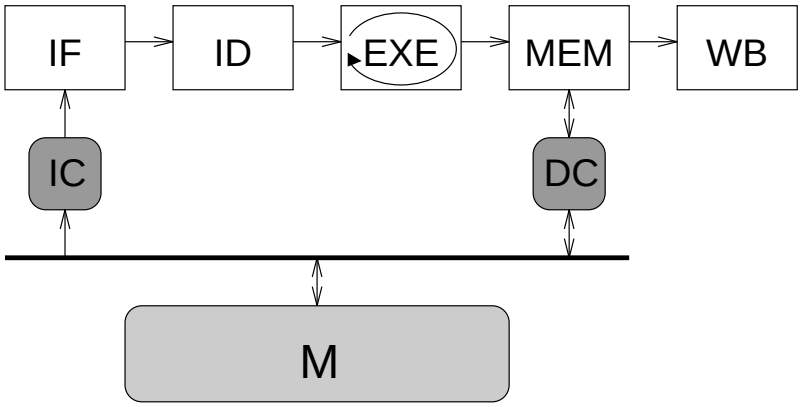
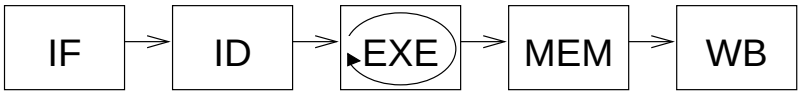
Advanced Pipeline Architectures



- So far we have considered only ideal pipelines where each stage will complete within one cycle in the absence of conflict with earlier pipeline instructions.
- Complications include:
  - Cache stalls
    e.g. Data cache read miss will stall all but WB stage.
  - Stage iteration
    e.g. SPARC v8 introduces instructions for signed and unsigned integer multiply. To avoid a very slow clock, we have iteration in the execute stage.

Advanced Pipeline Architectures

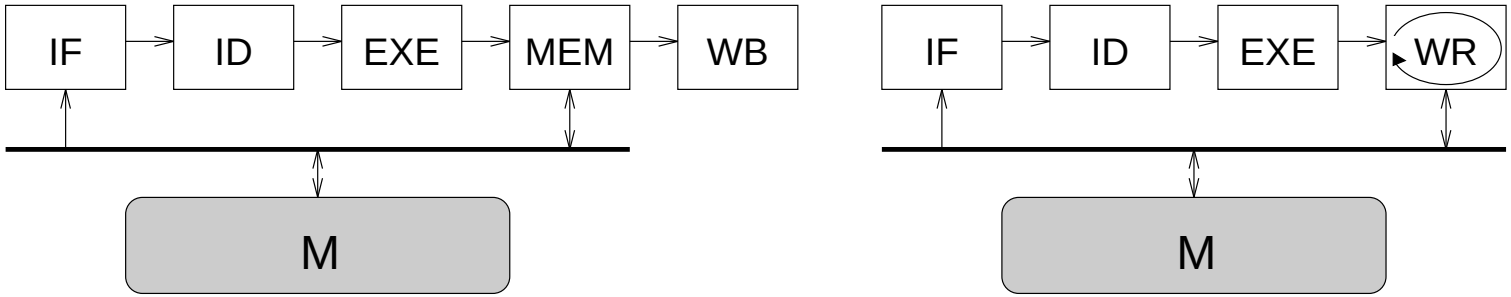


- Effect on pipeline of stage iteration during the execution of MULtiply:

- - -	IF	ID	EXE	MEM	WB									
MUL		IF	ID	EXE_1	EXE_2	EXE_3	EXE_4	EXE_5	EXE_6	EXE_7	EXE_8	MEM	WB	
- - -			IF	$^{ID}stall$	$^{ID}stall$	$^{ID}stall$	$^{ID}stall$	$^{ID}stall$	$^{ID}stall$	$^{ID}stall$	ID	EXE	MEM	WB
- - -				$^{IF}stall$	$^{IF}stall$	$^{IF}stall$	$^{IF}stall$	$^{IF}stall$	$^{IF}stall$	$^{IF}stall$	IF	ID	EXE	MEM
												IF	ID	EXE

Here the multiply is shown taking 8 cycles in the execute stage, the actual number of cycles will depend on the number of bits processed per iteration and the number of bits in the multiplier.

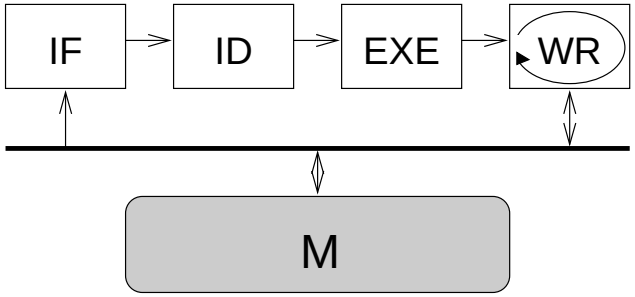
Princeton Pipeline Architectures



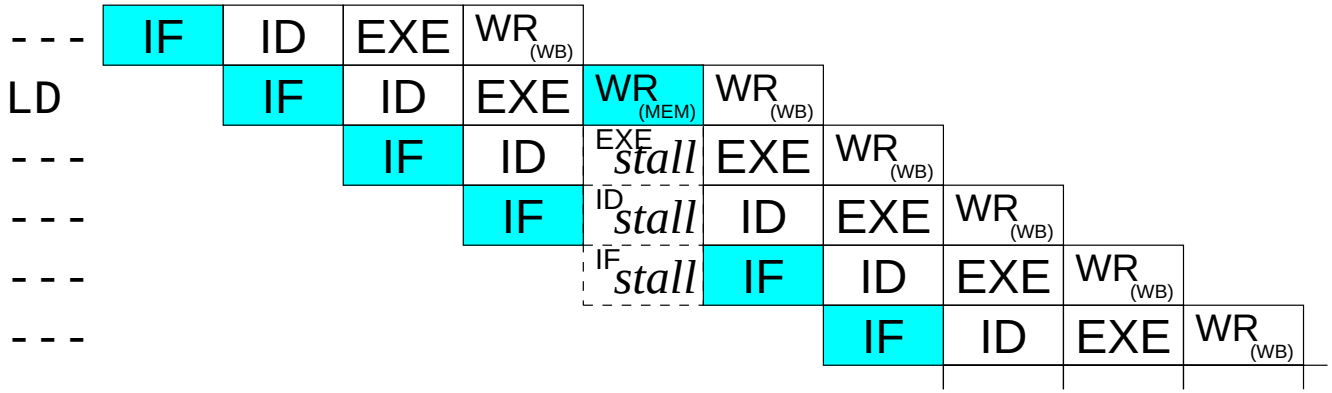
- A five stage Princeton architecture will never make use of all five stages, since the use of the MEM stage precludes the use of the IF stage.

- A simpler four stage implementation performs MEM and WB operations as required within the WRite stage. For a load instruction the WR stage will last for two cycles; WR(MEM) during which the rest of the pipeline is stalled and WR(WB) during which the register file is updated.

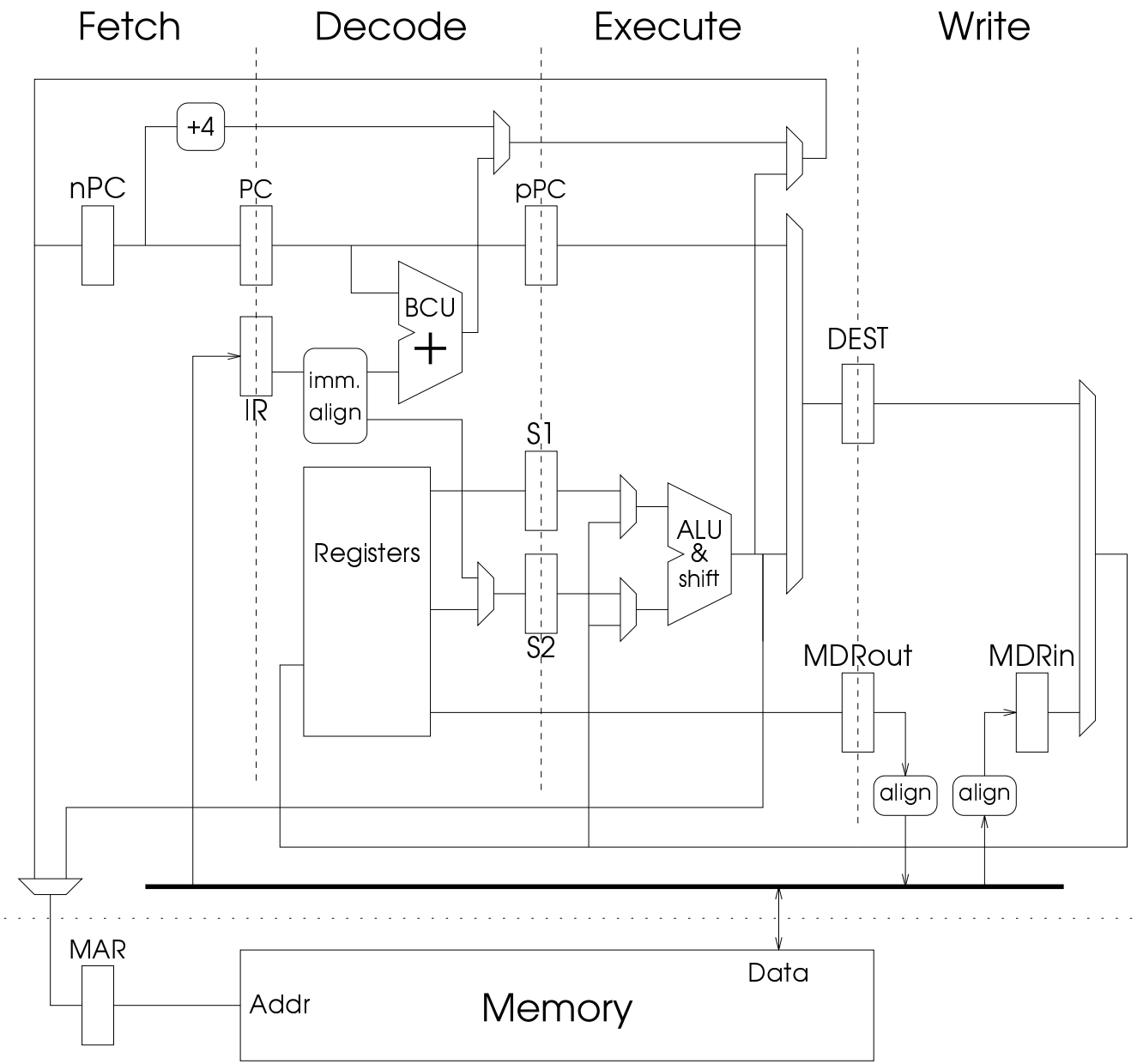
Advanced Pipeline Architectures



- Effect on pipeline of 5 cycle load in 4 stage architecture:

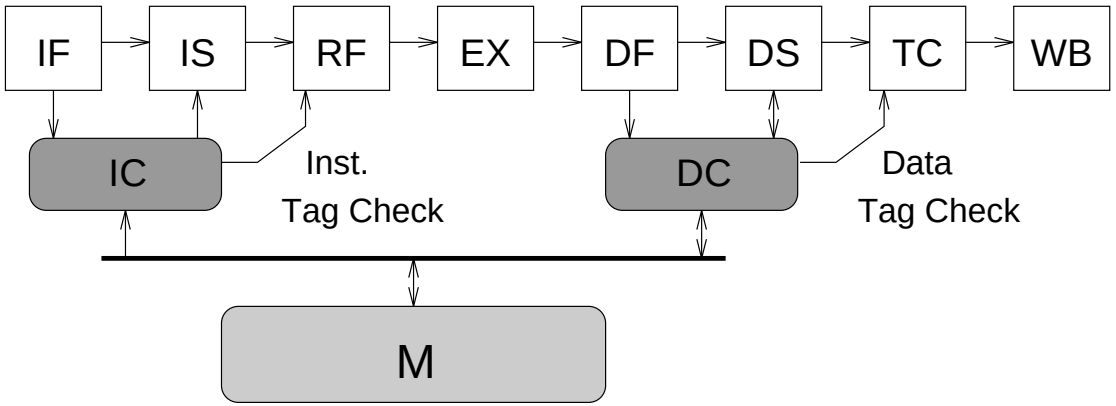


The LSI logic SPARC chip on the RISC Experimenter Board uses this system:


Fetch
Decode
Execute
Write
+4
nPC
PC
pPC
BCU
+
IR
imm.
align
DEST
S1
Registers
ALU
&
shift
S2
MDRout
MDRin
align
align
MAR
Addr
Memory
Data

SuperPipelined Architectures

MIPS R4000 8 stage pipeline[1]

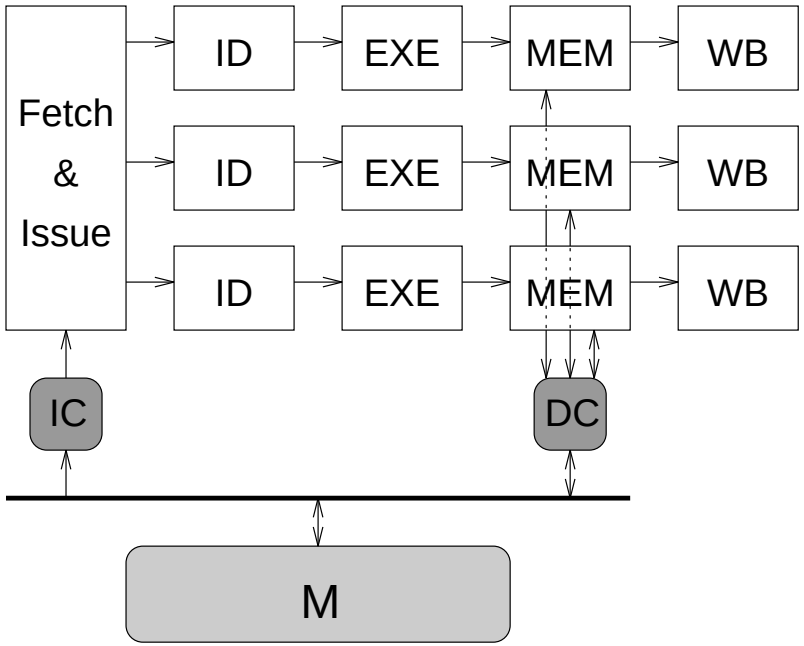


- Deeper pipe leads to faster clock.
- Deeper pipe leads to more hazards:
  - 2 cycle branch penalty (even with 1 cycle delayed branch)
  - 2 cycle load-use penalty
  - *Some pipelines have two execute stages incurring a single cycle define-use penalty*

[1] Hennessy & Patterson *Computer Architecture* pp201-209

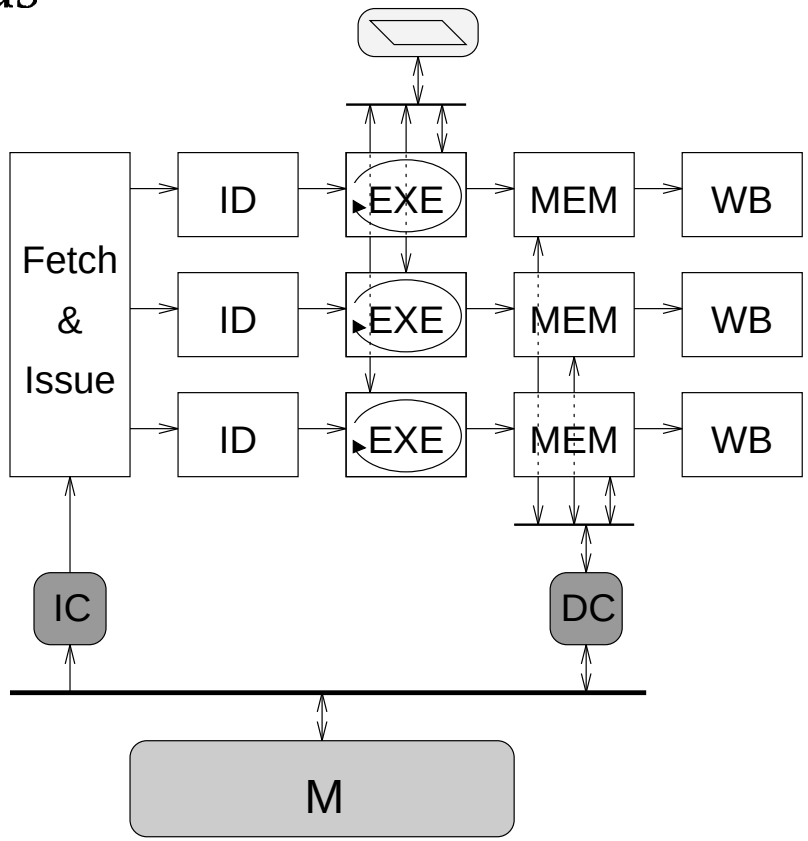
SuperScalar Pipeline Architectures

These processors have multiple execution units combined with an ability to launch multiple instructions in the same cycle. With suitable width buses to instruction and data caches these processors can achieve average CPI values of less than one.



SuperScalar Pipeline Architectures

Structural Hazards

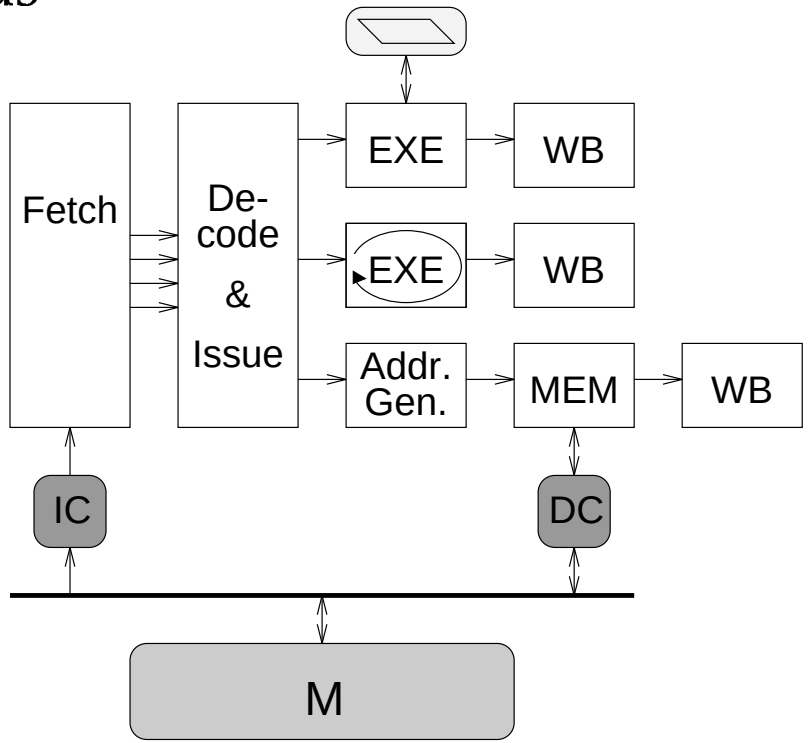


- Multiple general purpose pipelines compete for scarce resources.

Should we allow out-of-order completion?

SuperScalar Pipeline Architectures

Structural Hazards



- Multiple specialised pipelines with attached resources.
  - only certain types of instruction may be launched together

Should we allow out-of-order issue?

Data Hazards

- True Dependency

```
RAW:    ADD A,B,C
        MUL C,D,E
```

With more instructions in the system we have much more complicated dependencies including dependency on instructions in the same pipeline stage and possibly even dependency on instructions which have been overtaken.

  - Many processors enforce strict ordering of dependent instructions.
    Dependencies are detected before issue, preventing premature issue of instructions that would cause a hazard.

- False Dependencies

```
WAR:    SUB F,G,H          WAW:    XOR K,L,M
        ADD I,J,F                  AND N,O,M
```

These dependencies can be accommodated or removed using *register renaming*.

Control Hazards

- Branch Penalty

  A single cycle branch delay may cause up to $2n - 1$ instructions to be fetched in error in a machine which launches n instructions per cycle.

- Speculative Execution

  Branch prediction may lead to the speculative execution of instructions from the branch destination. We must ensure that the correct instructions are killed on branch

- Dynamic branch prediction

  Advanced systems predict the outcome of a branch instruction even before the instruction is fetched from the instruction cache, based on previous behaviour.

- Interrupts

  If we allow out of order completion of dependency-free instructions, what action should we take on interrupt?

  Should we consider each instruction that might generate a trap as a control transfer instruction on which all subsequent instructions depend?

Optimizing Compilers

Optimizing compilers are required to maximize the number of concurrent instructions.

- Instruction re-ordering

  Instruction may be re-ordered such that instructions arrive in batches which are concurrently executable.

  This manipulation is, of course, limited by the required program semantics.

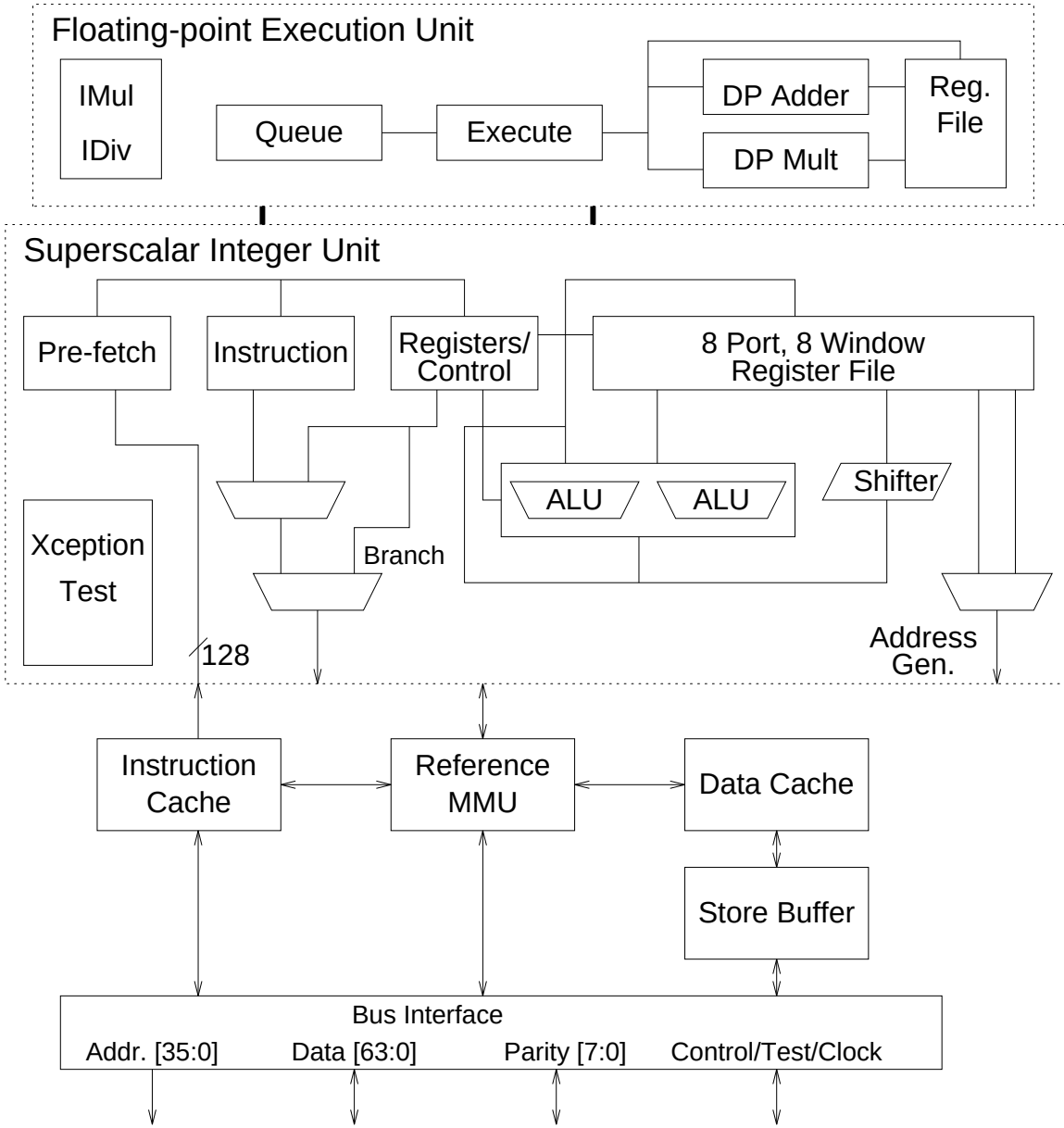
- Loop unrolling

  CPI value is severely limited by short runs of instructions between successful branches.

  Unrolling of loops increases the run length and may be used to convert a branch on X to a branch on $notX$ where X is the likely outcome.

SuperSPARC (Texas Instruments 1992)

- Launches up to 3 instructions per cycle.
  - average 0.75 CPI
- 50 MHz
- 64 bit floating point unit
- 20 kbyte (fully coherent) instruction cache
- 16 kbyte data cache
- 64 bit wide main memory access
- 3.1 million transistors


Floating-point Execution Unit
IMul
IDiv
Queue
Execute
DP Adder
DP Mult
Reg.
File
Superscalar Integer Unit
Pre-fetch
Instruction
Registers/
Control
8 Port, 8 Window
Register File
ALU
ALU
Shifter
Xception
Test
Branch
128
Address
Gen.
Instruction
Cache
Reference
MMU
Data Cache
Store Buffer
Bus Interface
Addr. [35:0]
Data [63:0]
Parity [7:0]
Control/Test/Clock

SuperPipelined SuperScalar Pipeline Architectures

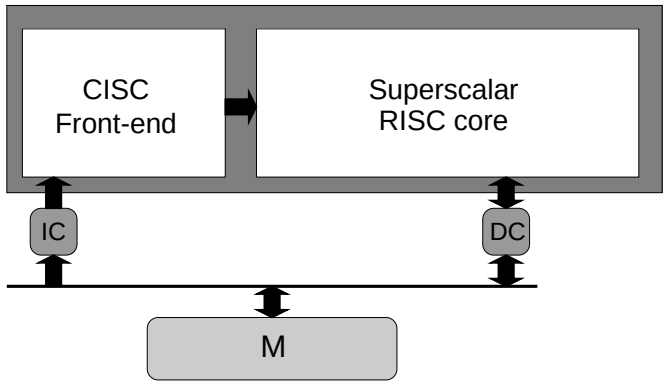
α 21164 (Digital Semiconductor 1995)

- 64 bit datapath 32 bit instructions
- 600 MHz
- 4 pipes
  - 2 integer @ 7 stages
  - 2 float @ 9 stages
- ordered execution of sets of up to 4 independent instructions aligned on 16 byte boundary
- 9.6 million transistors

α 21264 (Compaq 1998)

- Includes another 2 integer pipes (although issue rate remains at 4 instructions per cycle) and supports out of order instruction issue using 15.2 million transistors.
- 700MHz version is the current top performing μP 1/1/00

SuperScalar CISC – Pentium II/III



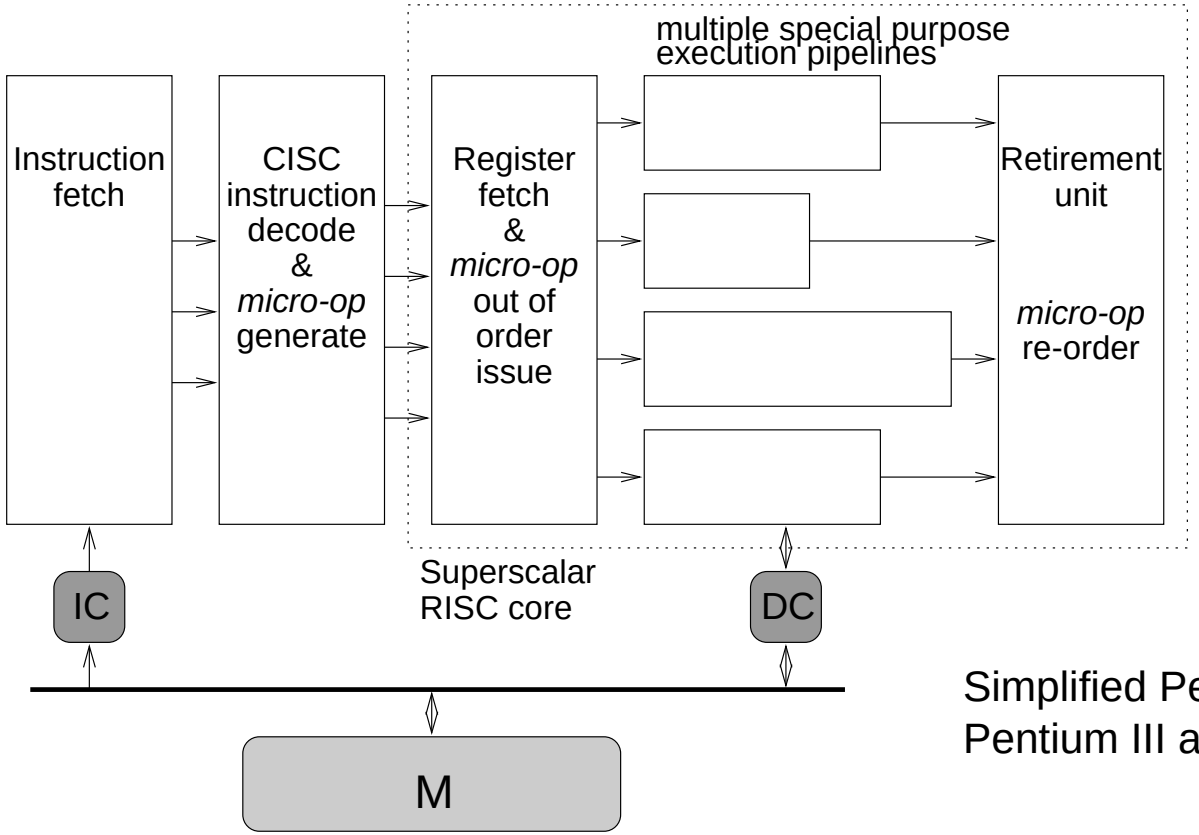
- With variable length, variable complexity instructions, CISC machines don't pipeline well.
- CISC instructions are converted to one or several *micro-ops* (uops).

  Each *micro-op* is 118 bits fixed format.
- Micro-ops are executed within a superscalar RISC core.

  The resulting pipeline is long and complicated (>14 stages) with most of the effort going into coping with the inherited instruction set.

  There are also problems with too few registers and with the floating point stack model causing an excess of memory accesses.

SuperScalar CISC – Pentium II/III



Simplified Pentium II / Pentium III architecture

- Micro-ops may be issued out of order - where operands are unavailable.
- Micro-op completion is ordered - avoiding problems with pipeline flush due to branch mis-prediction and exception handling.