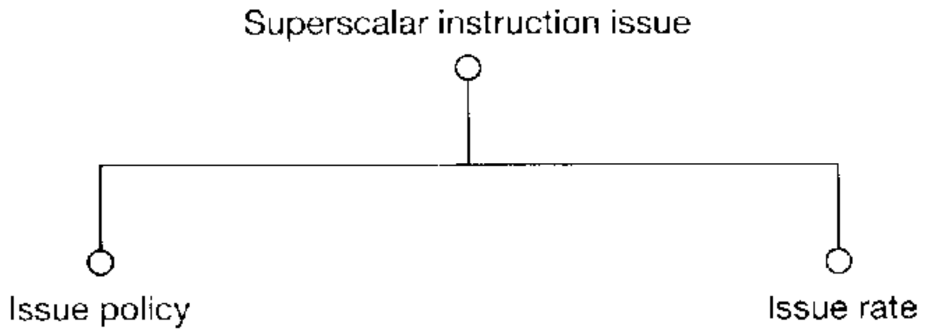
Instruction Issue

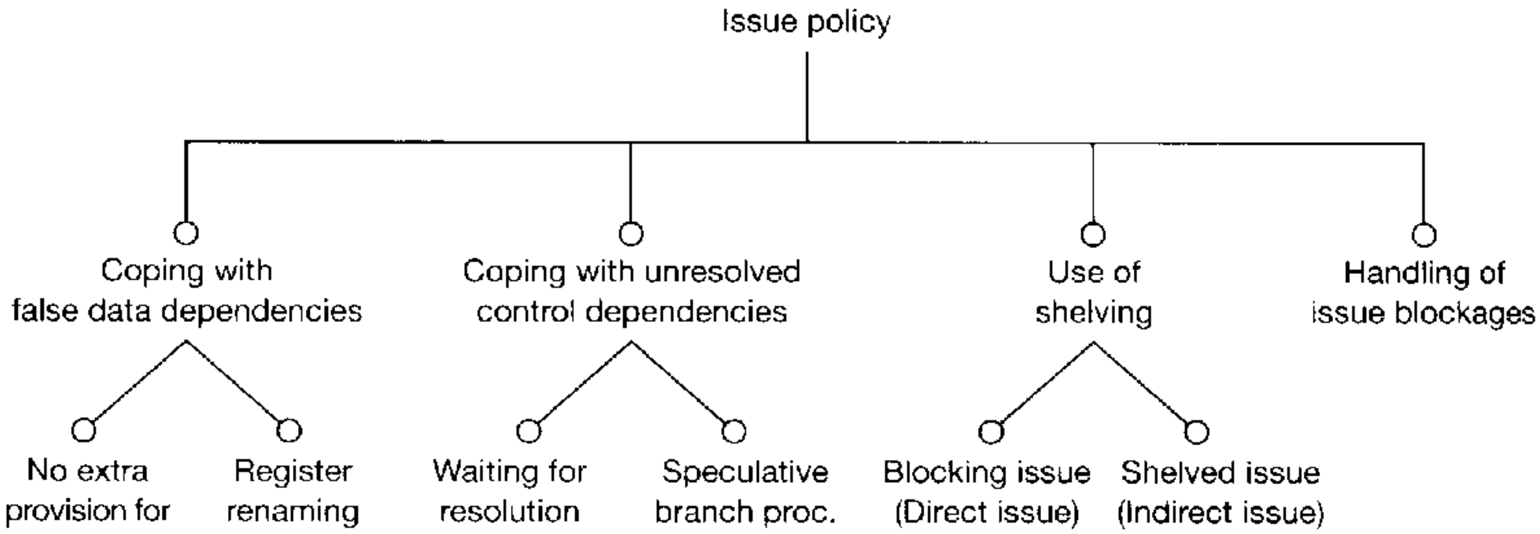
Superscalar instruction issue comprises two major aspects:

1. The *issue policy* specifies how dependencies are handled during the issue process.
2. The *issue rate*, on the other hand, specifes the maximum number of instructions a superscalar processor is able to issue in each cycle.



Instruction Issue Policies

The design space of issue policy is considerably complex but consists for 4 major aspects:



Coping with Dependencies

The first two aspects are concerned with how dependencies are coped with during instruction issue:

1. Whether or not the processor uses register renaming to *eliminate the false data register dependencies* which occur either between the instructions to be issued and those in execution, or among the instructions to be issued.

2. With respect to unresolved control dependencies, either the processor waits until the referenced condition becomes available or it employs *speculative execution* of the control transfer instructions by guessing the outcome.

Avoiding Issue blockages

- Decoded instructions to be issued are checked for dependencies. If dependencies exist between instructions they will cause issue blockages (*blocking issue*) unless Shelving is used to avoid such blockages.

- In the *blocking issue* (also called direct issue) approach, dependency checking takes place in the *issue window* which is comprised of the last n entries in the instruction buffer (where n is the *issue rate*).

- The use of the blocking issue mode heavily impedes issue performance.

Avoiding Issue blockages$_{(cont)}$

- *Shelving* (also called indirect issue) decouples instruction issue and dependency checking. This technique presumes special instruction buffers, often refered to as reservation stations, exist in front of functional units. Instructions are issued to shelving buffers essentially without making dependency checks between the instructions in the issue window and those executing. This kind of dependency checking is now delayed until a later step (*instruction dispatch*).

Handling Issue Blockages

Instruction issue blockages may occur with or without Shelving.

- When shelving is not used, any dependencies encountered in the issue window immediately block the issue of instructions.

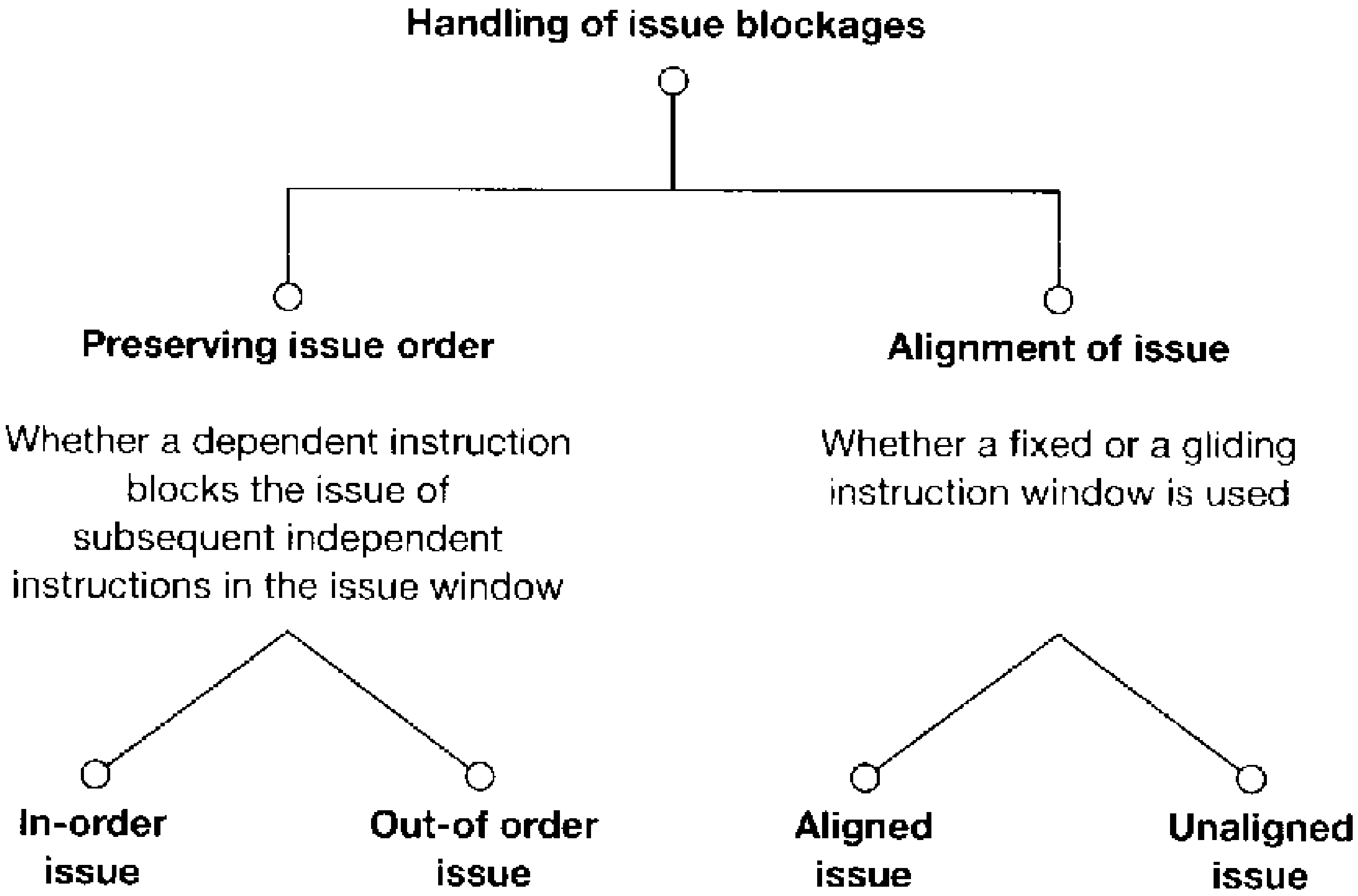
- In contrast, when shelving is used, issue blockages due to program dependencies are generally avoided. However, issue blockages may continue to occur due to certain hardware constraints.

Handling Issue Blockages$_{(cont)}$

The handling of blockages has two aspects:

1. *Preserving issue order* specifies whether a dependent instruction blocks the issue of subsequent independent instructions. Since this can severely impede performance, some superscalar processors allow certain types of instruction to be issued *out-of-order*.

2. *Alignment of instruction issue* determines whether a fixed or gliding issue window is used.

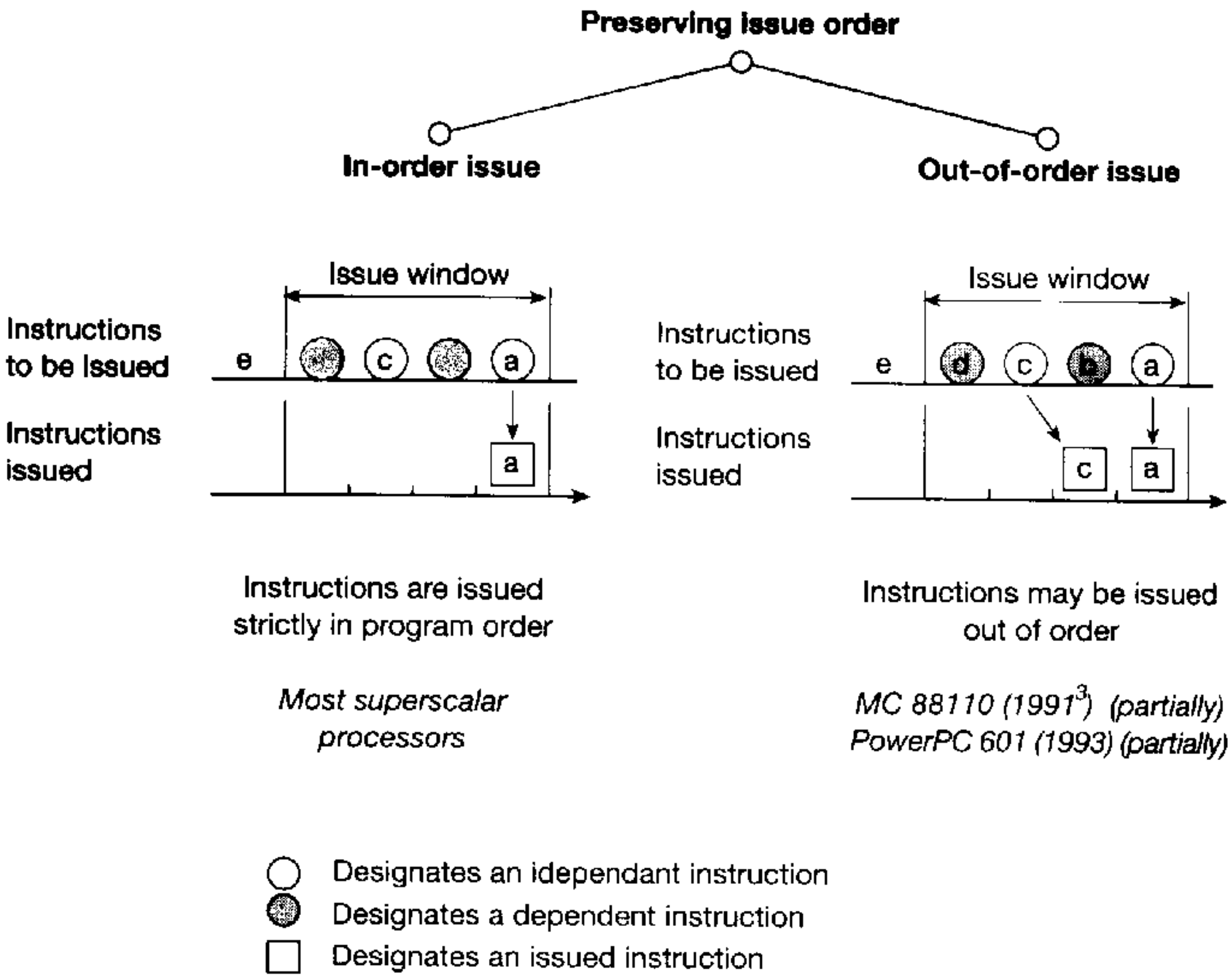

Handling of issue blockages
Preserving issue order
Alignment of issue
Whether a dependent instruction blocks the issue of subsequent independent instructions in the issue window
Whether a fixed or a gliding instruction window is used
In-order issue
Out-of order issue
Aligned issue
Unaligned issue

The Issue Order

There are two reason why few processors use out-of-order issue:

1. Preserving sequential consistency for out-of-order issue requires much more effort than for in-order issue.

2. Processors with shelving have almost no motivation for using out-of order issue since the issue of instructions rarely block due to resource constraints.


Preserving issue order
In-order issue
Out-of-order issue
Issue window
Instructions to be Issued
e
c
a
Instructions issued
a
Instructions are issued strictly in program order
Most superscalar processors
Issue window
Instructions to be issued
e
d
c
b
a
Instructions issued
c
a
Instructions may be issued out of order
MC 88110 (1991[3]) (partially)
PowerPC 601 (1993) (partially)
Designates an idependant instruction
Designates a dependent instruction
Designates an issued instruction

The Alignment of Instruction Issue

The second aspect of handling issue blockages determines the alignment of instruction issue, which refers to whether instructions are issued from a fixed or gliding window.

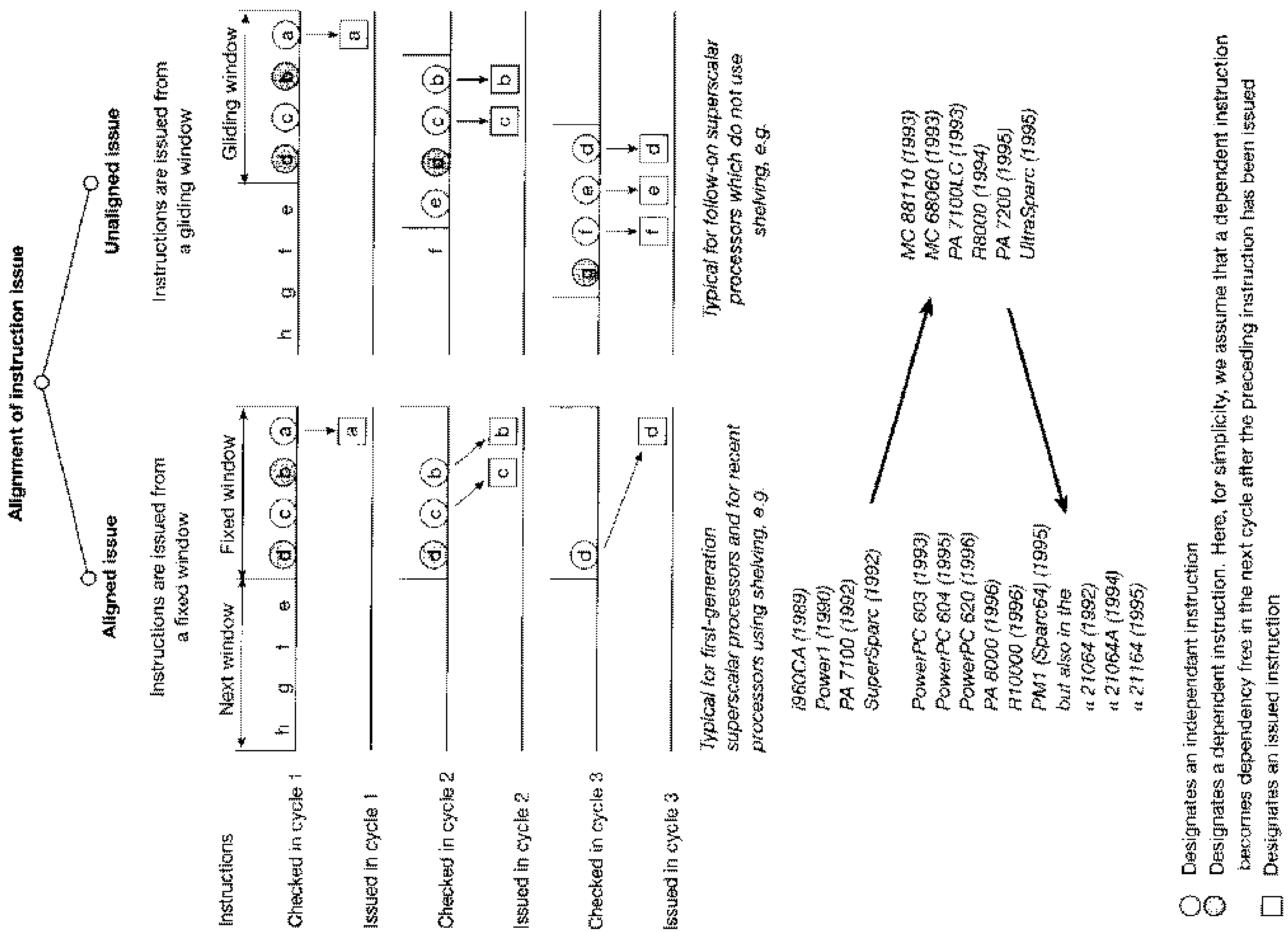
[By its nature, issue alignment is revelant only for superscalar processors.]

- Aligned instruction issue uses a *fixed window* which means that no instructions in the next window are considered as candidates for issue until all of the instructions in the current window have been issued. (Typical in first generation superscalar processors). In the blocking issue mode, aligned instruction issue considerably reduces the effective issue rate.

The Alignment of Instruction Issue$_{(cont)}$

- In a number of subsequent architectures which still used the blocking issue mode, unaligned instruction issue was introduced. A *gliding window* with a size equal to the issue rate is employed and where, in each cycle, all of the instructions are checked for dependencies. After the issuing of independent instructions, the window is refilled.

Instruction alignment can be summed up by saying that unaligned instruction issue was introduced to increase the performance of superscalar processors employing the blocking issue mode. However, with the application of shelving unaligned instruction issue has lost its rationale, and the most recently introduced processor have gone back to using aligned issue.

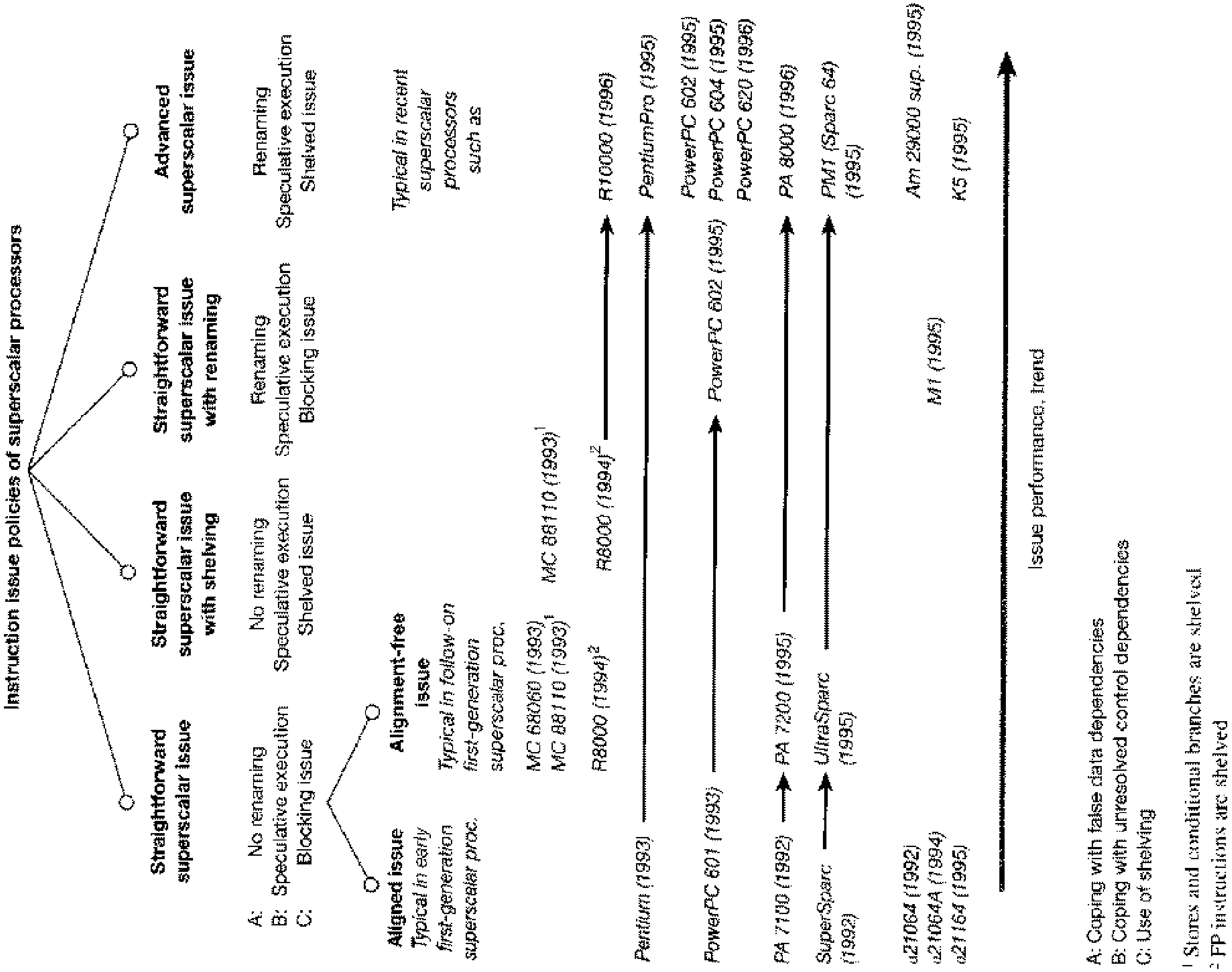

Alignment of instruction issue
Aligned issue
Instructions are issued from a fixed window
Unaligned issue
Instructions are issued from a gliding window
Instructions
Next window
Fixed window
Gliding window
h g f e d c b a
Checked in cycle 1
Issued in cycle 1
Checked in cycle 2
Issued in cycle 2
Checked in cycle 3
Issued in cycle 3
Typical for first-generation superscalar processors and for recent processors using shelving, e.g.
i960CA (1989)
Power1 (1990)
PA 7100 (1992)
SuperSparc (1992)
PowerPC 603 (1993)
PowerPC 604 (1995)
PowerPC 620 (1996)
PA 8000 (1996)
R10000 (1996)
PM1 (Sparc64) (1995)
but also in the
α 21064 (1992)
α 21064A (1994)
α 21164 (1995)
Typical for follow-on superscalar processors which do not use shelving, e.g.
MC 88110 (1993)
MC 68060 (1993)
PA 7100LC (1993)
R8000 (1994)
PA 7200 (1995)
UltraSparc (1995)
Designates an independant instruction
Designates a dependent instruction. Here, for simplicity, we assume that a dependent instruction becomes dependency free in the next cycle after the preceding instruction has been issued
Designates an issued instruction

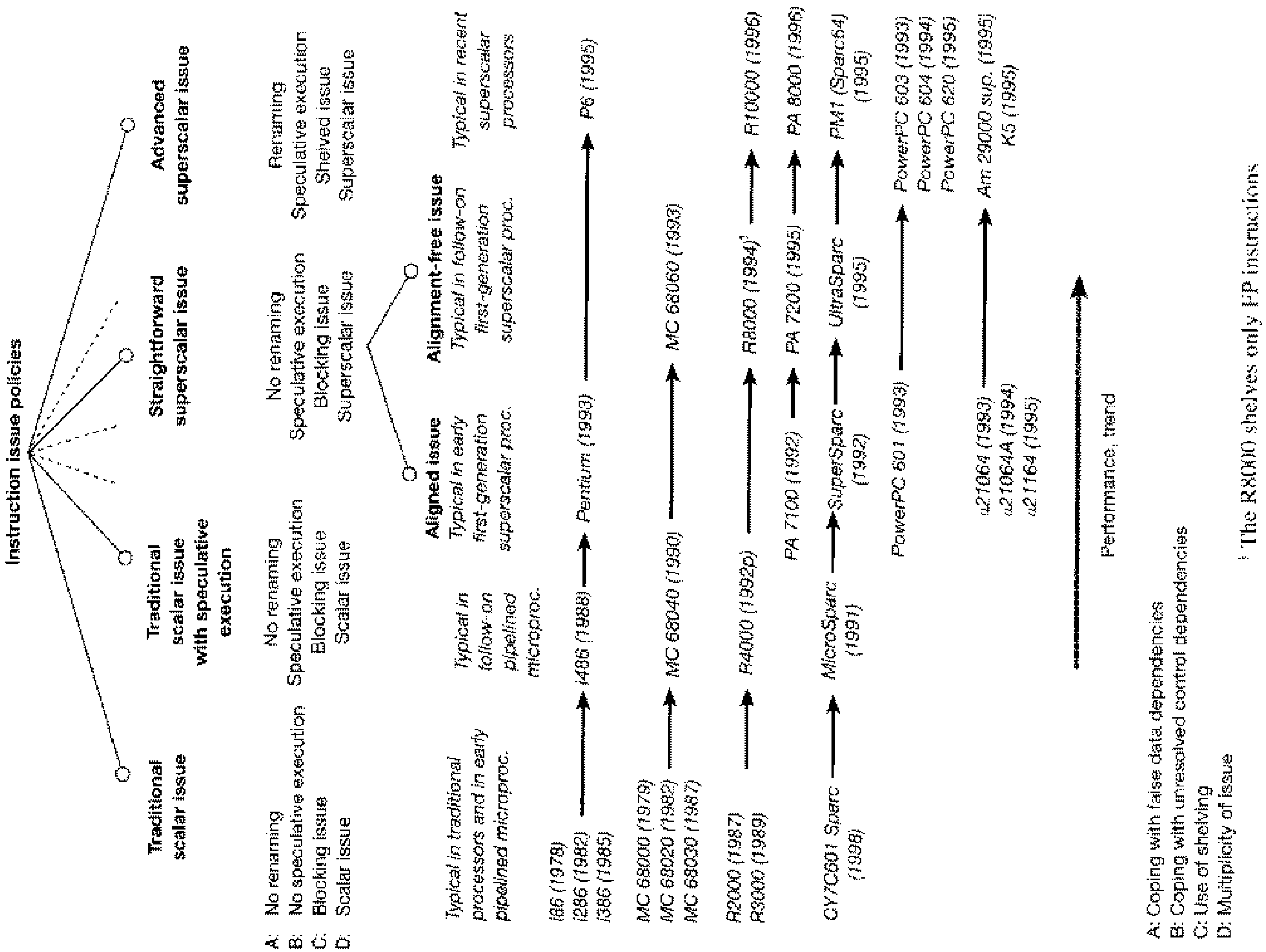
Which Issue Policy should be used?

The design space of instruction issue policies span four major aspects and the handling of blockages covers a further two. Thus, there are five aspects, each a binary choice, resulting in $2^5 = 32$ possible issue policies. Not all policies are of equal importance and therefore less important aspects can be neglected, e.g. issue order can be ignored since most processors (scalar and superscalar) employ in-order issue.

- Scalar processors really just have to consider three issue aspects: whether renaming, speculative execution and shelving are employed or not.

- Superscalars consider issue alignment in addition to the three basic aspects used in scalar processors.


Instruction issue policies of superscalar processors
Straightforward superscalar issue
Straightforward superscalar issue with shelving
Straightforward superscalar issue with renaming
Advanced superscalar issue
A: No renaming
B: Speculative execution
C: Blocking issue
No renaming
Speculative execution
Shelved issue
Renaming
Speculative execution
Blocking issue
Renaming
Speculative execution
Shelved issue
Aligned issue
Typical in early first-generation superscalar proc.
Alignment-free issue
Typical in follow-on first-generation superscalar proc.
Typical in recent superscalar processors such as
MC 68060 (1993)
MC 88110 (1993)
MC 88110 (1993)
R8000 (1994)
R8000 (1994)
R10000 (1996)
Pentium (1993)
PentiumPro (1995)
PowerPC 601 (1993)
PowerPC 602 (1995)
PowerPC 602 (1995)
PowerPC 604 (1995)
PowerPC 620 (1996)
PA 7100 (1992)
PA 7200 (1995)
PA 8000 (1996)
SuperSparc (1992)
UltraSparc (1995)
PM1 (Sparc 64) (1995)
α21064 (1992)
α21064A (1994)
α21164 (1995)
M1 (1995)
Am 29000 sup. (1995)
K5 (1995)
Issue performance, trend
A: Coping with false data dependencies
B: Coping with unresolved control dependencies
C: Use of shelving
1 Stores and conditional branches are shelved
2 FP instructions are shelved



Instruction issue policies
Traditional scalar issue
Traditional scalar issue with speculative execution
Straightforward superscalar issue
Advanced superscalar issue
A: No renaming
B: No speculative execution
C: Blocking issue
D: Scalar issue
No renaming
Speculative execution
Blocking issue
Scalar issue
No renaming
Speculative execution
Blocking issue
Superscalar issue
Renaming
Speculative execution
Shelved issue
Superscalar issue
Aligned issue
Alignment-free issue
Typical in traditional processors and in early pipelined microproc.
Typical in follow-on pipelined microproc.
Typical in early first-generation superscalar proc.
Typical in follow-on first-generation superscalar proc.
Typical in recent superscalar processors
i86 (1978)
i286 (1982)
i386 (1985)
i486 (1988)
Pentium (1993)
P6 (1995)
MC 68000 (1979)
MC 68020 (1982)
MC 68030 (1987)
MC 68040 (1990)
MC 68060 (1993)
R2000 (1987)
R3000 (1989)
R4000 (1992p)
R8000 (1994)¹
R10000 (1996)
PA 7100 (1992)
PA 7200 (1995)
PA 8000 (1996)
CY7C601 Sparc (1988)
MicroSparc (1991)
SuperSparc (1992)
UltraSparc (1995)
PM1 (Sparc64) (1995)
PowerPC 601 (1993)
PowerPC 603 (1993)
PowerPC 604 (1994)
PowerPC 620 (1995)
α21064 (1993)
α21064A (1994)
α21164 (1995)
Am 29000 sup. (1995)
K5 (1995)
Performance, trend
A: Coping with false data dependencies
B: Coping with unresolved control dependencies
C: Use of shelving
D: Multiplicity of issue
¹ The R8000 shelves only FP instructions

Issue Rate

The issue rate (as known as the *degree of superscalarity*) refers to the maximum number of instructions a superscalar processor can issue in the same cycle.

Superscalar operation may be implemented by issuing two, three, or more instructions in each cycle. Evidently, a higher issue rate offers a higher performance potential; however, its implementation requires more complex circuitry.