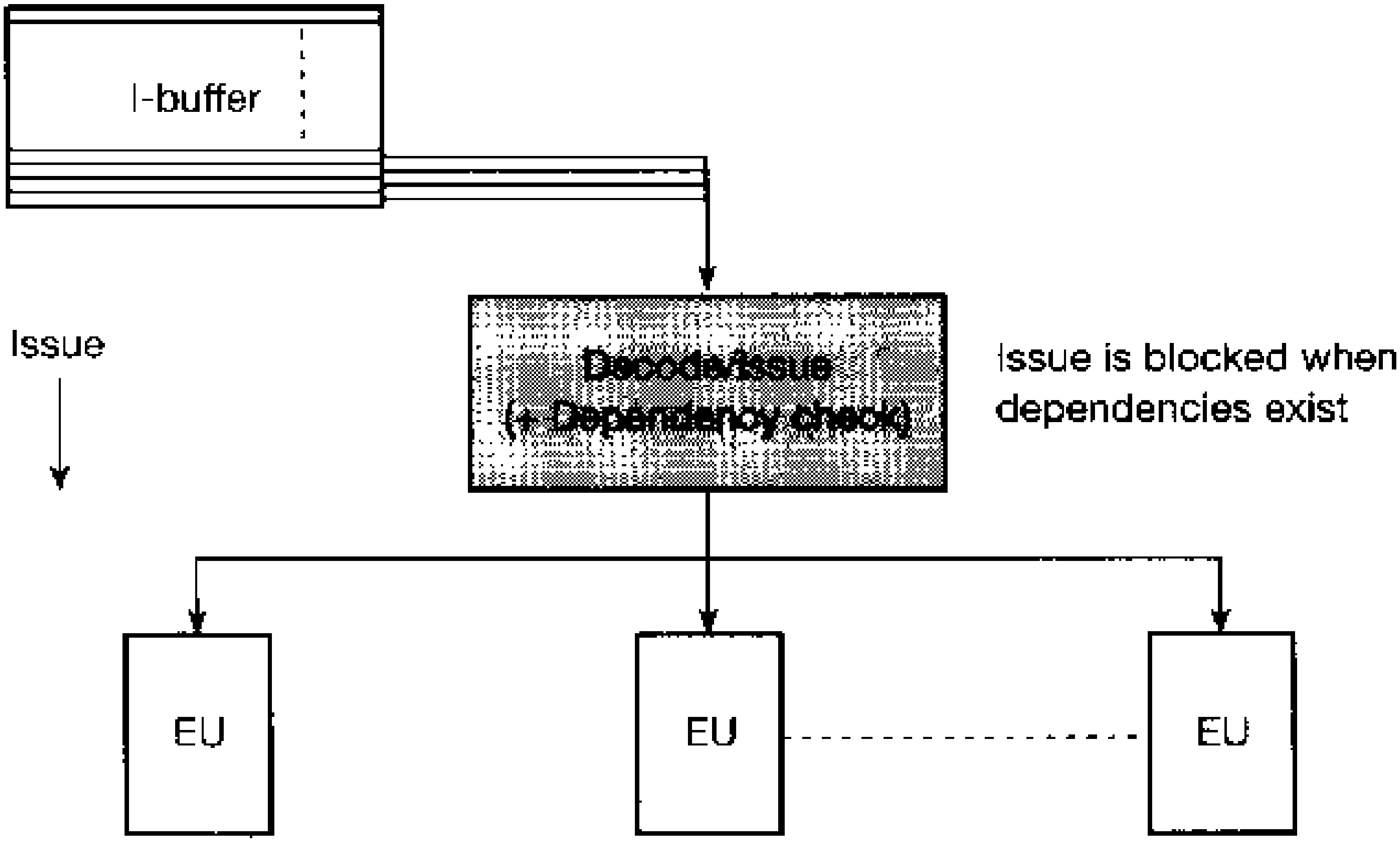
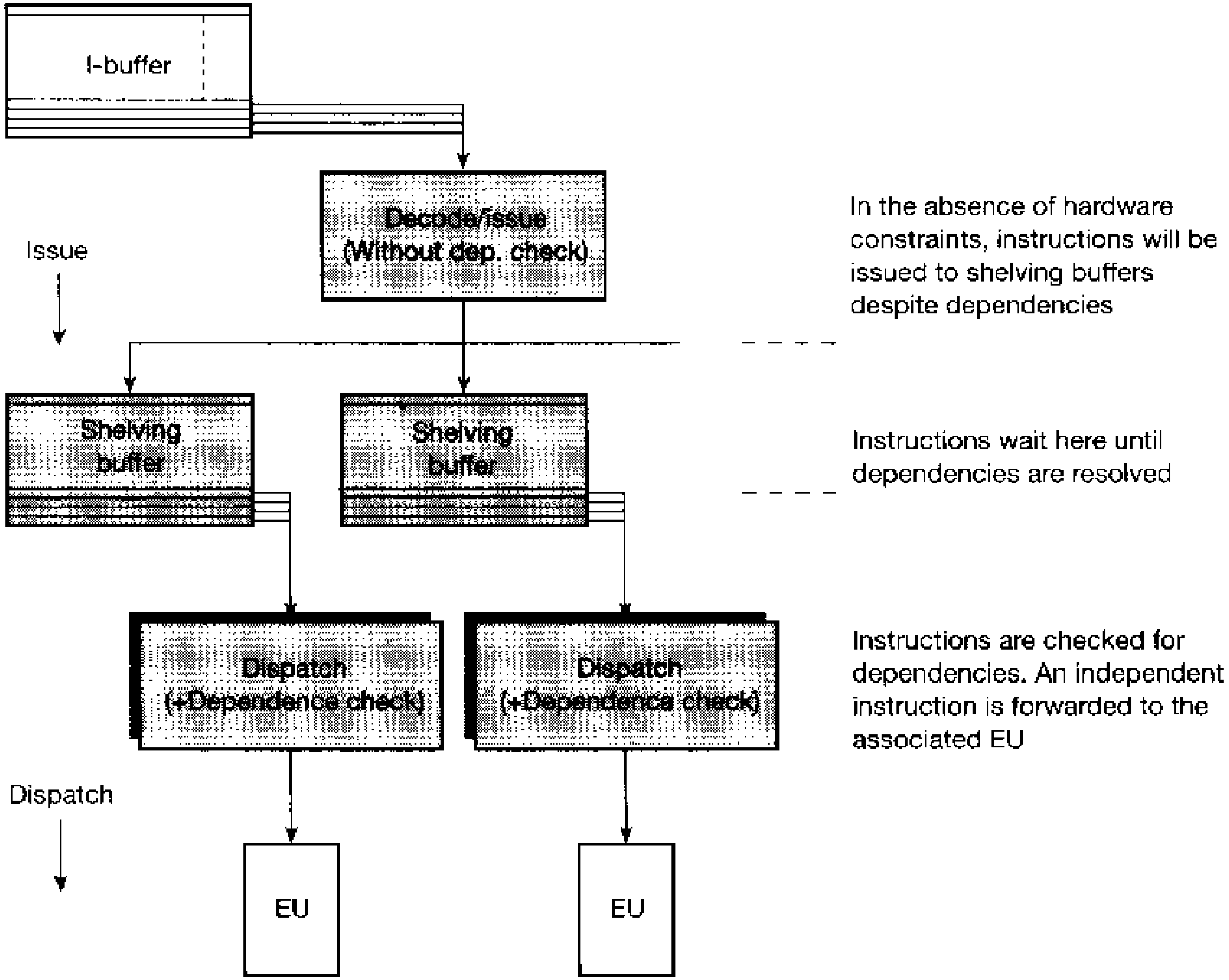
Shelving

Shelving is an advanced issue mode which is employed to eliminate issue blockages due to dependencies. Shelving makes use of dedicated instruction buffers, called *shelving buffers*, in front of each functional or execution unit (EU).

- Shelving decouples dependency checking from the instruction issue. More precisely, instructions are issued to sheving buffers without any checks for data or control dependencies or for busy EUs.
- Nevertheless, certain constraints can restrain the processor to issue fewer instructions in a cycle than its issue rate.

[Processors which use shelving usually employ in-order, aligned issue.]


I-buffer
Issue
Decode/issue
(+ Dependency check)
Issue is blocked when dependencies exist
EU
EU
EU


I-buffer
Decode/issue
(Without dep. check)
Issue
Shelving buffer
Shelving buffer
Dispatch
(+Dependence check)
Dispatch
(+Dependence check)
Dispatch
EU
EU
In the absence of hardware constraints, instructions will be issued to shelving buffers despite dependencies
Instructions wait here until dependencies are resolved
Instructions are checked for dependencies. An independent instruction is forwarded to the associated EU

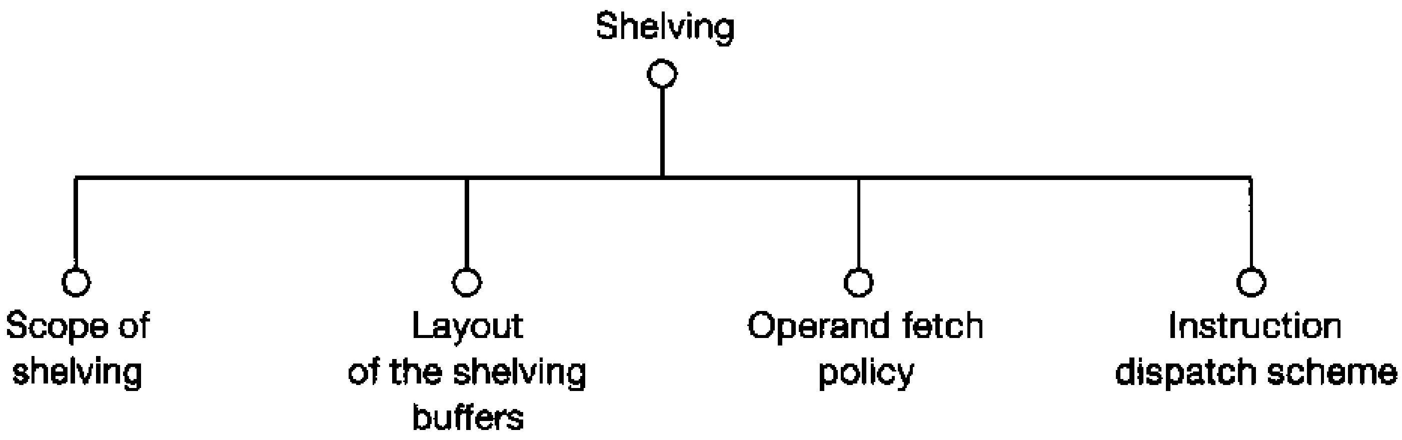
Shelving

Processors which use shelving predominantly employ the advanced superscalar issue policy. This means shelving is usually employed in connection with:

- *speculative execution* which eliminates execution blockages due to unresolved control dependencies;
- *register renaming* which removes false dependencies.

Therefore, only true dependencies (RAWs) can prevent instructions held in shelving buffers from being executed.

Shelving is also used in connection with a method of retaining *sequential consistency* called *instruction reordering*. Since the number of instruction available for execution is increased by the number held in shelving buffer, shelving makes processing more distributed.



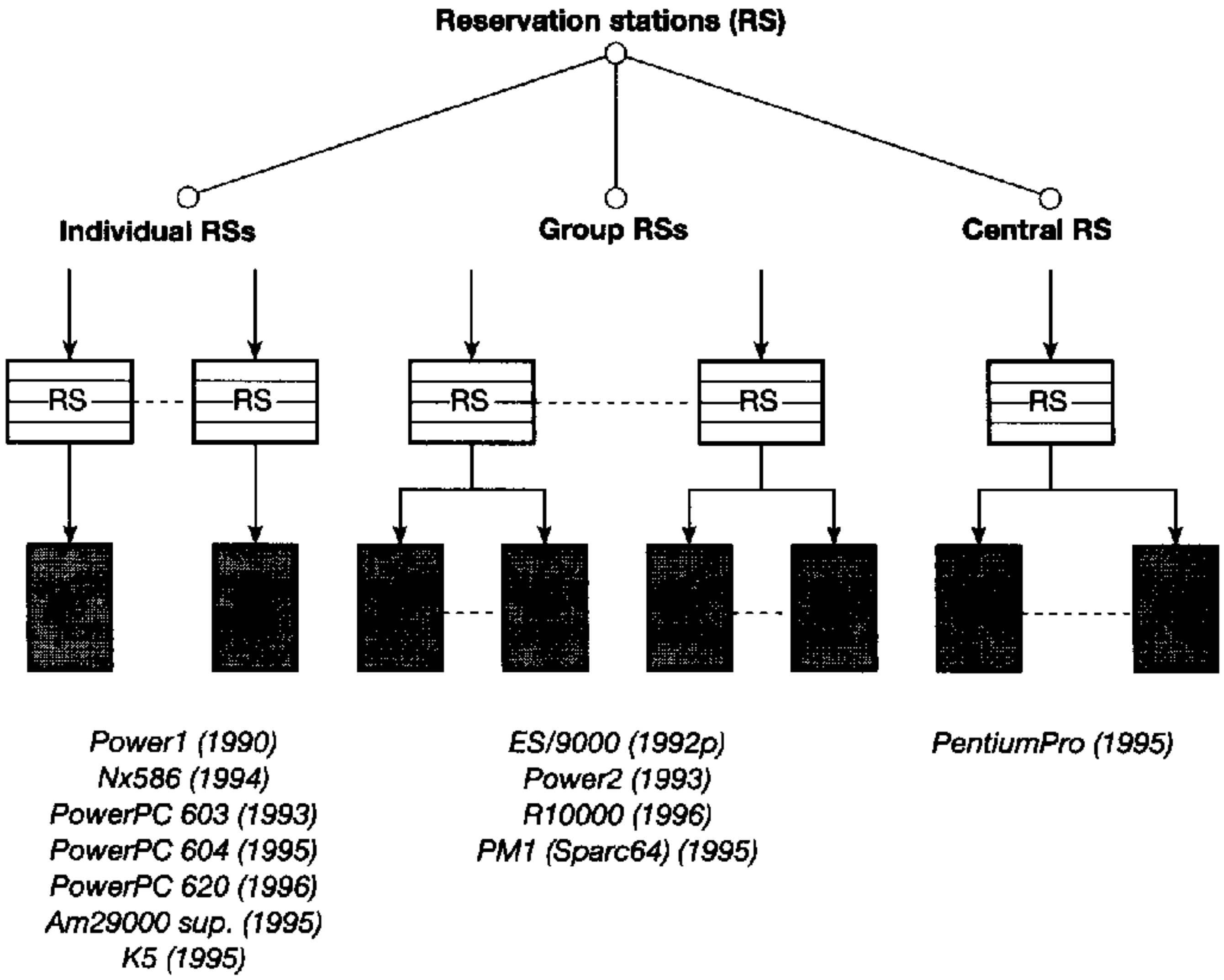
Scope of Shelving

Partial shelving is when shelving is restricted to one or to a few instruction types. This is an incomplete solution to the problem of eliminating issue blockages due to dependencies, so most recent superscalar processor use *full shelving*.

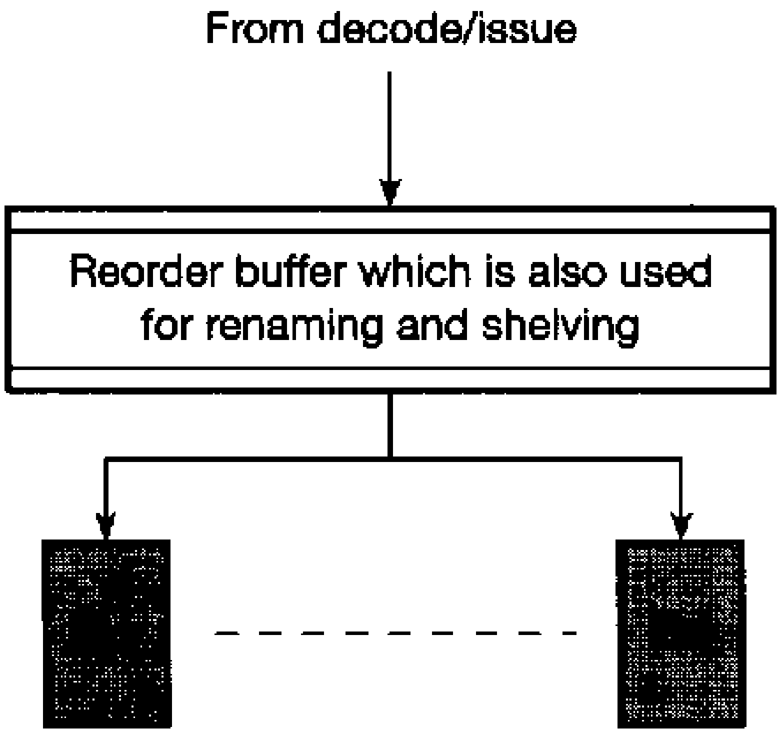
Layout of Shelving Buffers

Shelving buffers hold instructions from their issue until they are forwarded to an EU. They have three major aspects:

- their type – the most common implementation is *standalone shelving buffers* or *reservation stations* which are used exclusively for shelving, or *combined buffers* which are used for shelving, renaming and reordering;
- their capacity – the number of shelving buffer entries (typically between 15 and 50 in recent processors); and
- the number of read and write ports.


Reservation stations (RS)
Individual RSs
Group RSs
Central RS
RS
RS
RS
RS
RS
Power1 (1990)
Nx586 (1994)
PowerPC 603 (1993)
PowerPC 604 (1995)
PowerPC 620 (1996)
Am29000 sup. (1995)
K5 (1995)
ES/9000 (1992p)
Power2 (1993)
R10000 (1996)
PM1 (Sparc64) (1995)
PentiumPro (1995)

Combined buffer for shelving, renaming and reordering



Lightning (1991p)

Also designed as DRIS (**D**eferred scheduling, **R**egister renaming, **I**nstruction **S**helve)

Operand Fetch Policies

Closely connected with shelving is the policy governing how processors fetch operands:

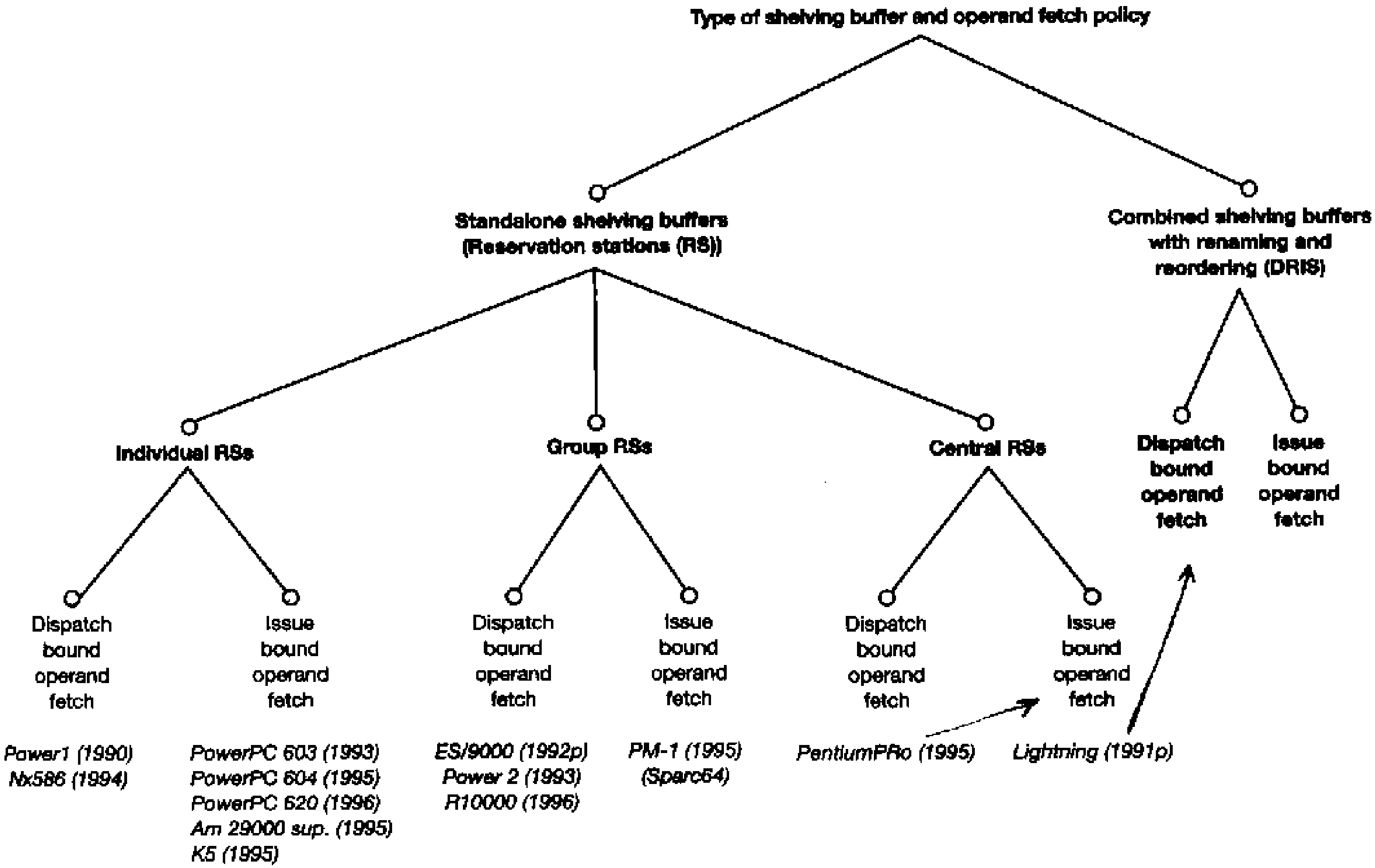
- The *issue bound operand fetch policy* means that operands are fetched during instruction issue and therefore the shelving buffers have to provide entries long enough to hold the source operands.

- The *dispatch bound operand fetch policy* fetches operands during dispatching and so the shelving buffers need only contain short register identifiers instead of the long operands.

Comparing Operand Fetch Policies

- If operands are fetched during instruction issue, the register file must be able to supply operands for all of the issued instructions at the same time. Assuming two source operands per instruction, the register file requires twice as many read ports as the maximum issue rate.
- If operands are fetched during instruction dispatch, ideally the number of read ports should equal the dispatch rate. However, the maximum dispatch rate is usually higher than the maximum issue rate.

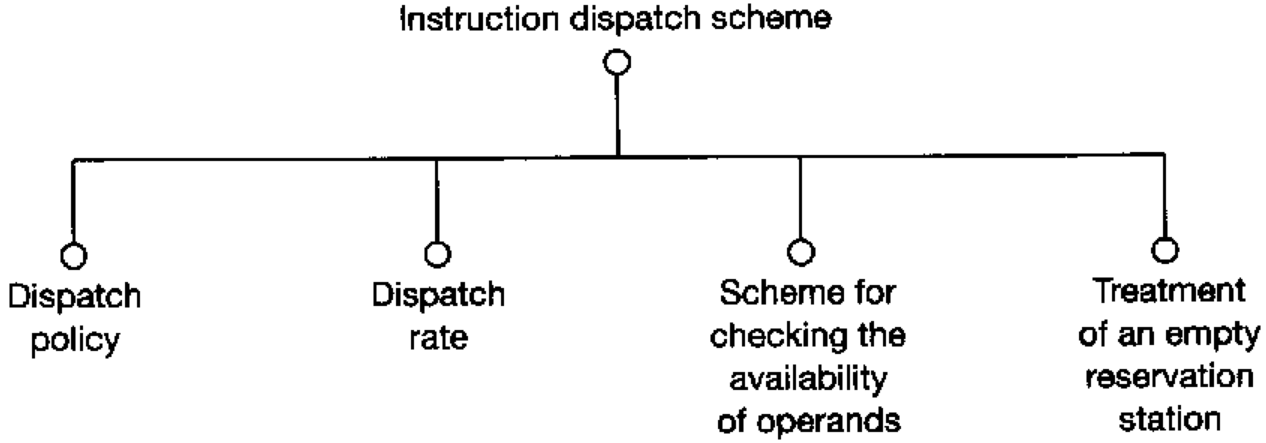
The dispatch bound policy has two advantages over the issue bound one: the critical issue/decode path is shorter; and the shelving buffer less complex.


Type of shelving buffer and operand fetch policy
Standalone shelving buffers
(Reservation stations (RS))
Combined shelving buffers
with renaming and
reordering (DRIS)
Individual RSs
Group RSs
Central RSs
Dispatch bound operand fetch
Issue bound operand fetch
Dispatch bound operand fetch
Issue bound operand fetch
Dispatch bound operand fetch
Issue bound operand fetch
Dispatch bound operand fetch
Issue bound operand fetch
Power1 (1990)
Nx586 (1994)
PowerPC 603 (1993)
PowerPC 604 (1995)
PowerPC 620 (1996)
Am 29000 sup. (1995)
K5 (1995)
ES/9000 (1992p)
Power 2 (1993)
R10000 (1996)
PM-1 (1995)
(Sparc64)
PentiumPRo (1995)
Lightning (1991p)

Instruction Dispatch Schemes

Instruction dispatch can be broken down into two basic tasks:

1. *Scheduling the instructions* held in a particular reservation station for execution; and
2. *Disseminating* the scheduled instruction(s) to their allocated EU(s).



Dispatch Policy

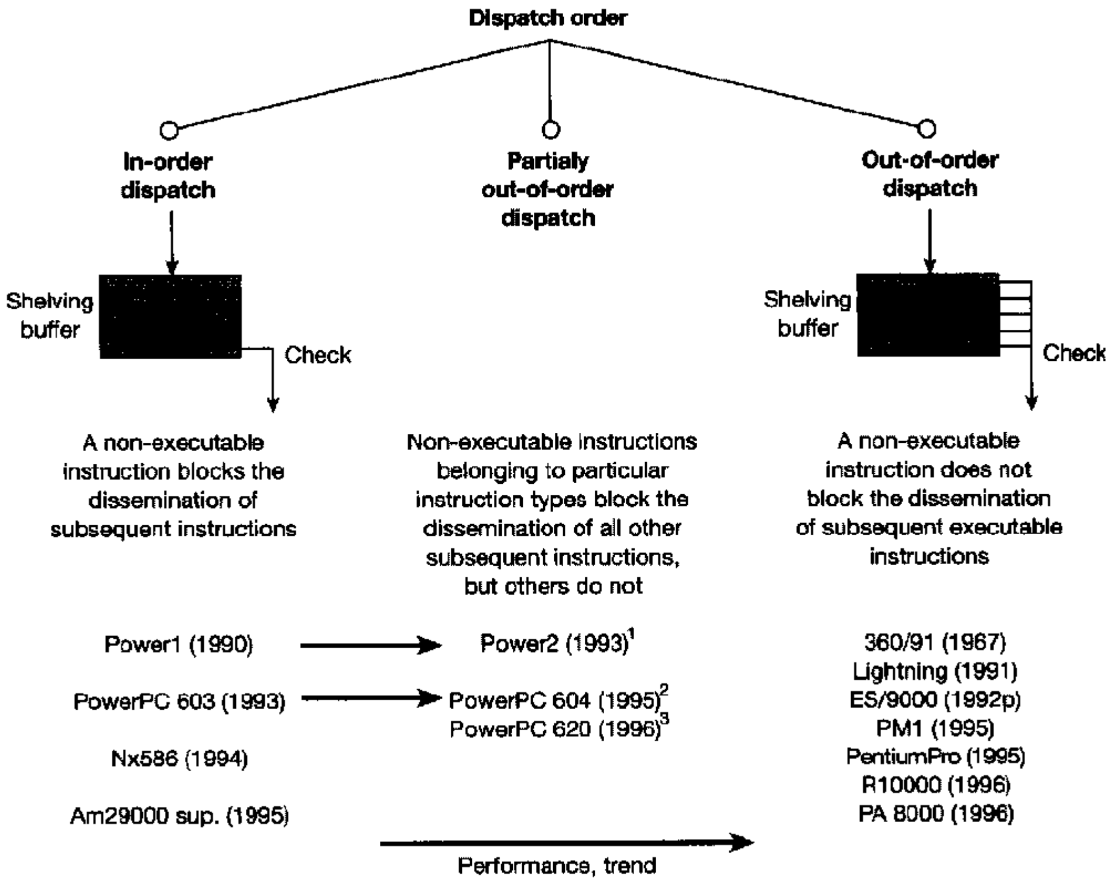
Dispatch policy may be considered as a scheduling policy consisting of the following three components:

- The *Selection Rule* specifies when instructions are executable. Assuming register renaming, speculative execution and the availability of the EU, those instructions whose operands are available are executable (this is known as the *dataflow principle of operation*). In other cases, the rule must include further dependency checks.

Dispatch Policy$_{(cont)}$

- The *Arbitration Rule* is needed when there are more executable instructions than can be disseminated in the next cycle. Most processors use a rule that chooses "older" instructions before "newer" ones.

- The *Dispatch Order* determines whether a non-executable instruction prevents all subsequent instructions from being dispatched.
(Similar to issue order.)



[1] In the Power2, only a single pending (not executable) FP instruction can be skipped.

[2] Out-of-order dispatch from the three integer reservation stations, but In-order dispatch from the Branch, Load/Store and FP reservation stations.

[3] Out-of-order dispatch from the three integer and Load/Store reservation stations, In-order dispatch from the Branch and FP reservation stations.

Dispatch Rate

Unlike individual reservation stations, a group or central reservation station, or a DRIS, must be capable of dispatching more than one instruction in each cycle.

The number of instructions that can be dispatched from each of the reservation stations, or from the DRIS, per cycle is the *dispatch rate.*

Ideally, a shelving buffer must be capable of dispatching one instruction to any EU connected to it in each cycle. Obviously this is easier to achieve for group stations with two to three EUs than for a central station or a DRIS with a considerable number.

Why Higher Dispatch Rates

The motivation for setting higher dispatch rate than issue rates is the following. Shelving allows the maximum number of instructions, as specified by the issue rate, to be issued in most cycles. However, complex instructions such as division, square root calculation etc., cannot be started immediately one after the other. Thus, for a balance of instruction issue and dispatch, the processor should be capable of dispatching more instructions to avaliable EUs than it issues to the shelving buffers per cycle.

Checking Availability of Operands

Irrespective of the operand fetch policy, the availability of operands needs to be checked in two scenarios:

1. when they are fetched from the register file, a scheme is needed to check whether the requested contents are available; and

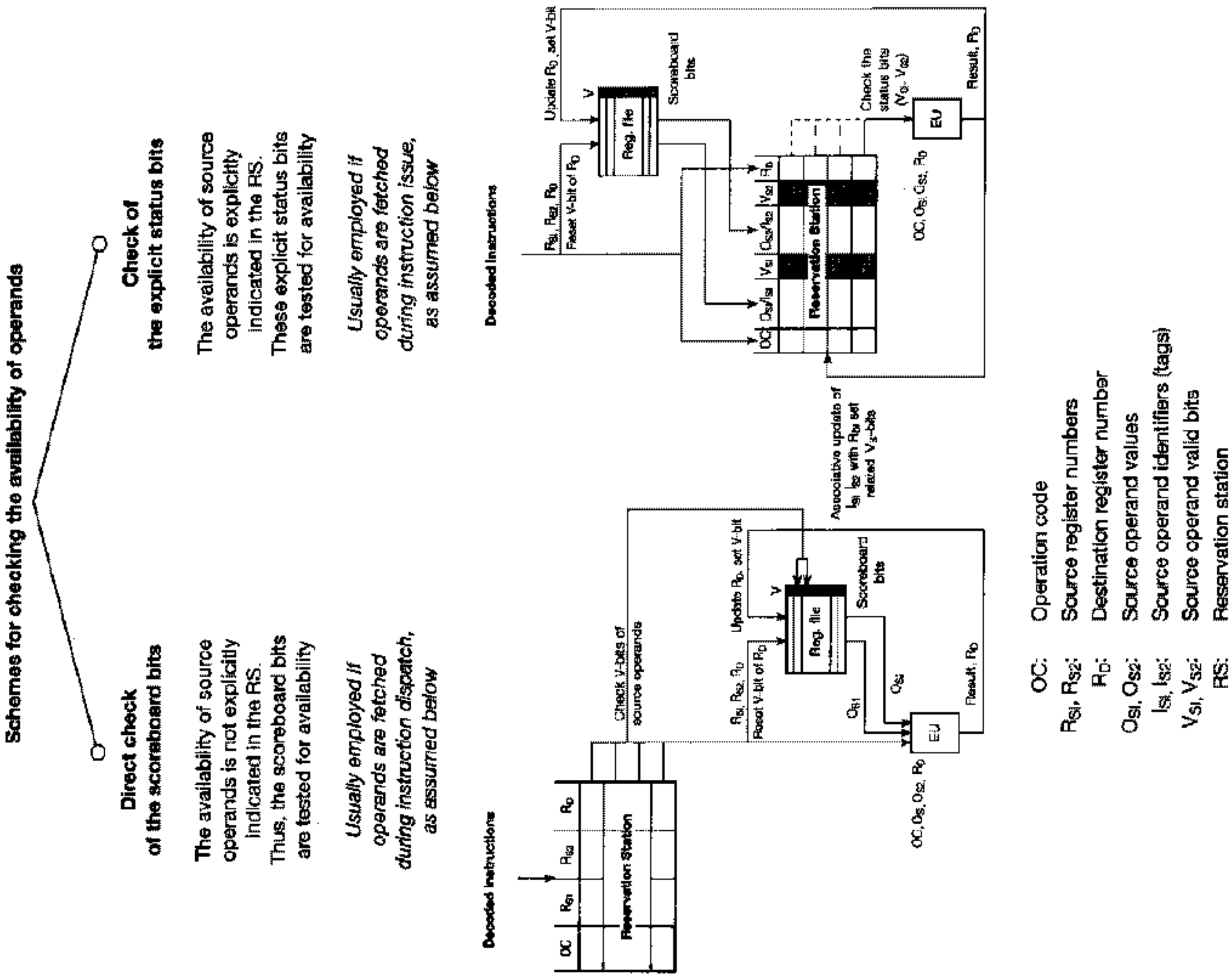
2. during instruction dispatch, a scheme is needed to check whether all the operands of the instructions held in the shelving buffers are available.

A technique known as *scoreboarding* is generally used to indicate the availability of register operands.

Scoreboarding

A scoreboard is actually a status register consisting of one-bit entries. Each bit may be envisioned as a one-bit extension of the corresponding register which indicated whether or not the associated data is valid.

- When an instruction is issued or dispatched, the scoreboard bit of the corresponding destination register is reset. This indicates to all subsequent instructions requesting the contents of the register in question that its value is not yet available.
- Once the instruction has executed and the result is written into the register, the corresponding scoreboard bit is set. Now the register contents are available to all requesting instructions.


Schemes for checking the availability of operands
Direct check
of the scoreboard bits
The availability of source operands is not explicitly indicated in the RS. Thus, the scoreboard bits are tested for availability
Usually employed if operands are fetched during instruction dispatch, as assumed below
Check of
the explicit status bits
The availability of source operands is explicitly indicated in the RS. These explicit status bits are tested for availability
Usually employed if operands are fetched during instruction issue, as assumed below
Decoded instructions
Reservation Station
Check V-bits of source operands
Reg. file
Scoreboard bits
EU
Result, R_D
Associative update of I_S1, I_S2 with R_D, set related V_S-bits
Check the status bits (V_S1, V_S2)
OC: Operation code
R_S1, R_S2: Source register numbers
R_D: Destination register number
O_S1, O_S2: Source operand values
I_S1, I_S2: Source operand identifiers (tags)
V_S1, V_S2: Source operand valid bits
RS: Reservation station

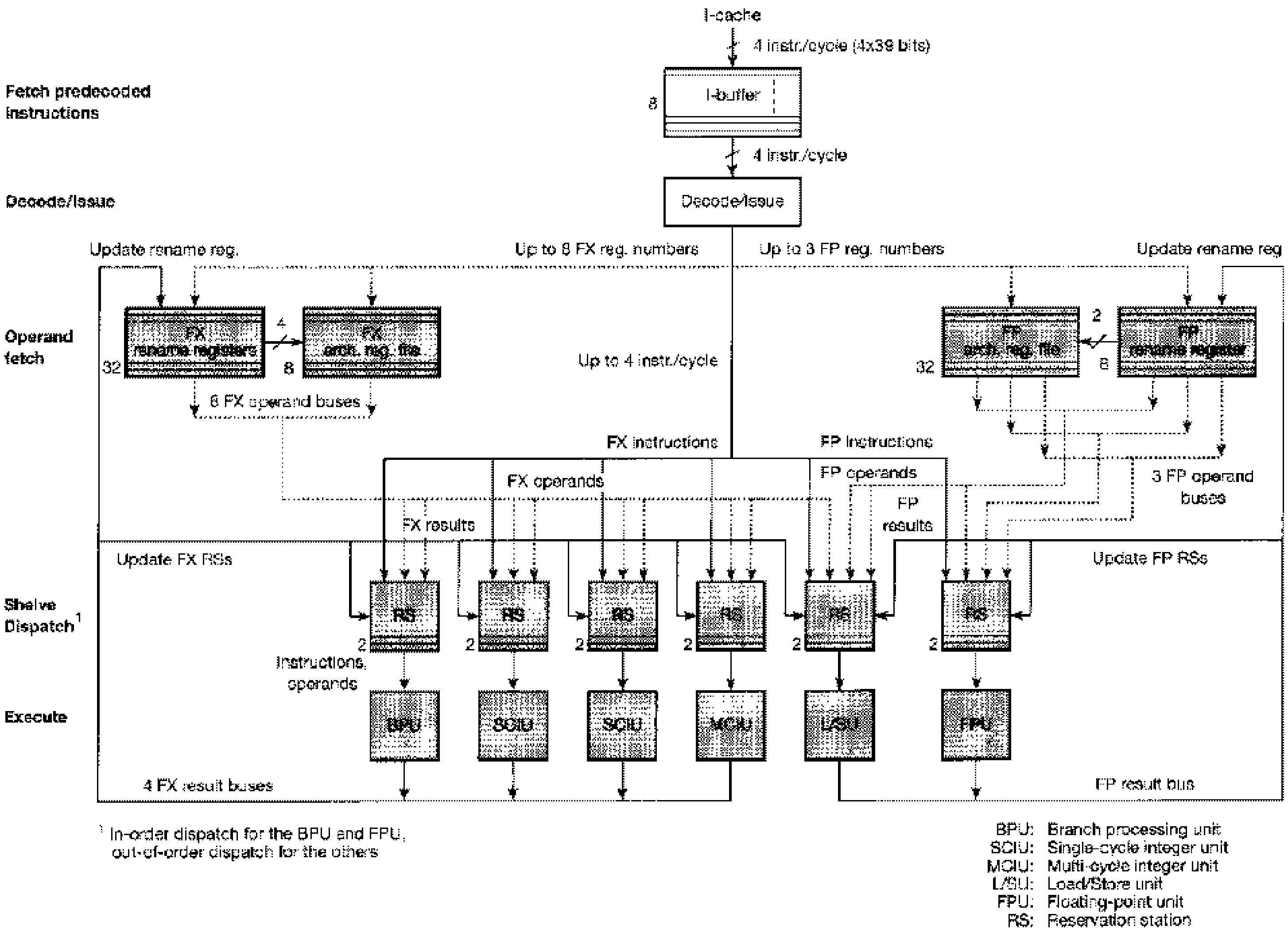
Extensions to the Basic Scenario

- If *multiple reservation stations* exist, their updating must be done *globally*, i.e. all results must be forwarded, along with their identifier, to any reservation station.
  - As many result buses are required as there are EUs operating in parallel.
- In the case of *split register files*, operand fetching and updating are separated into distinct fixed (FX) and floating point (FP) parts.

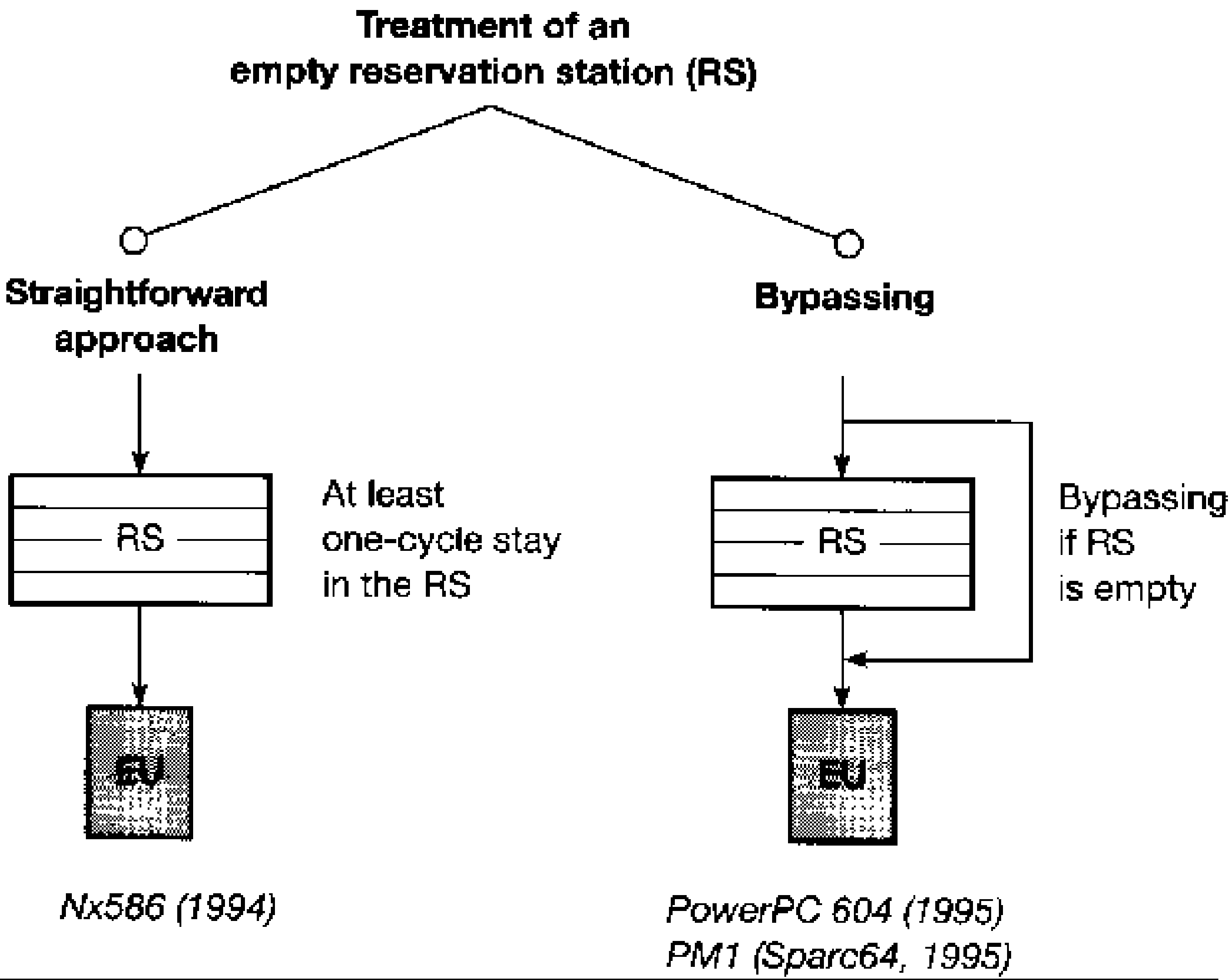
Extensions to the Basic Scenario$_{(cont)}$

- Multiple availability checks are required for group or central reservation stations.

- For out-of-order dispatch, register renaming is required and all instructions in the reservation station have to be checked in parallel. Furthermore, in the direct checking scheme the scoreboard bits belonging to the destination registers of the issued instructions have to be reset during instruction issue rather than instruction dispatch.



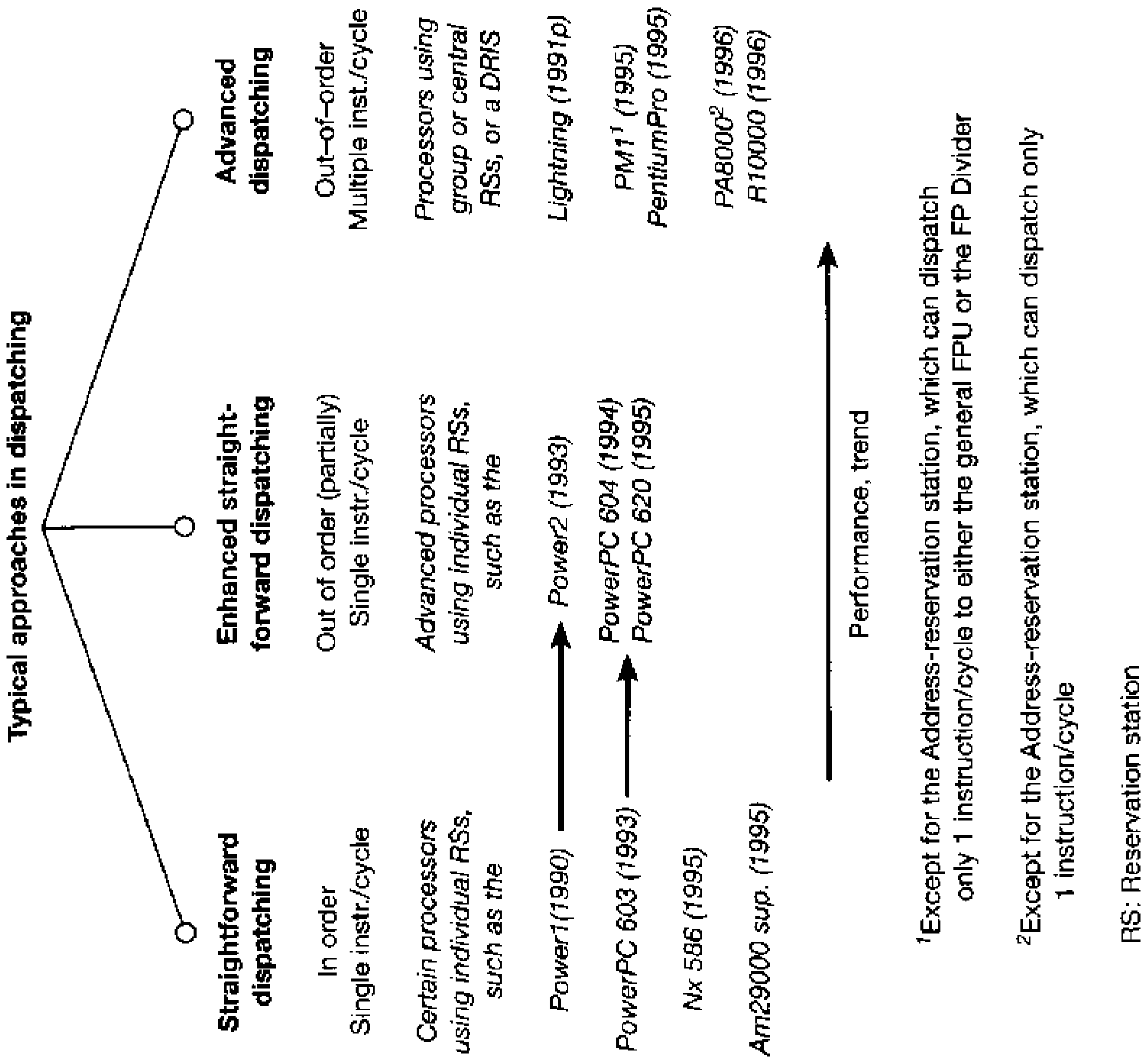
Internal data paths of the PowerPC 604.


Treatment of an
empty reservation station (RS)
Straightforward
approach
Bypassing
RS
At least
one-cycle stay
in the RS
RS
Bypassing
if RS
is empty
EU
EU
Nx586 (1994)
PowerPC 604 (1995)
PM1 (Sparc64, 1995)

Typical Dispatch Schemes

Typical solutions used for the various aspects of the design space are:

- Most recent superscalar processors employ register renaming and speculative execution. Thus, the dataflow selection rule is used most often, with the oldest instruction chosen if arbitration is needed.
- Operand availability is checked using one of the two methods described earlier depending on when the operands are fetched.
- An empty reservation station is usually bypassed.
- With respect to the *dispatch order* and *dispatch rate*, three typical approaches can be identified.


Typical approaches in dispatching
Straightforward dispatching
In order
Single instr./cycle
Certain processors using individual RSs, such as the
Power1(1990)
PowerPC 603 (1993)
Nx 586 (1995)
Am29000 sup. (1995)
Enhanced straight-forward dispatching
Out of order (partially)
Single instr./cycle
Advanced processors using individual RSs, such as the
Power2 (1993)
PowerPC 604 (1994)
PowerPC 620 (1995)
Advanced dispatching
Out–of–order
Multiple inst./cycle
Processors using group or central RSs, or a DRIS
Lightning (1991p)
PM1¹ (1995)
PentiumPro (1995)
PA8000² (1996)
R10000 (1996)
Performance, trend
¹Except for the Address-reservation station, which can dispatch only 1 instruction/cycle to either the general FPU or the FP Divider
²Except for the Address-reservation station, which can dispatch only 1 instruction/cycle
RS: Reservation station

Straightforward Dispatching

- Instructions are dispatched one at a time.
- An instruction that is not yet executable *blocks* further dispatching.
- Some processors which use individual reservation stations employ this simple but inefficient approach.
- The efficiency can be increased by introducing out-of-order dispatching for particular reservation stations. This is called *enhanced straightforward dispatching*.

Enhanced Straightforward Dispatching

- Subsequent instructions can bypass an instruction that is not yet executable.

- Obviously this results in a performance increase over the basic approach.

- This approach is typically employed in advanced processors using individual reservation stations.

Advanced Dispatching

- Is the most powerful.
- It allows *out-of-order dispatch* and is capable of forwarding *multiple instructions* per cycle to available EUs.
- This policy is employed in processors with group reservation stations, a central reservation station or with a DRIS.