Internal distribution code:

(A) [-] Publication in OJ
(B) [-] To Chairmen and Members
(C) [-] To Chairmen
(D) [X] No distribution

Datasheet for the decision of 29 June 2021

Case Number: T 0058/16 - 3.5.06

Application Number: 08790429.8

Publication Number: 2325744

IPC: G06F9/34, G06F9/40, G06F9/30

Language of the proceedings: EN

Title of invention:
PROCESSING UNIT

Applicant:
Fujitsu Limited

Headword:
Arithmetic processing unit/FUJITSU

Relevant legal provisions:
EPC Art. 56, 84, 83, 123(2)

Keyword:
Claims - clarity (yes) - support in the description (yes)
Sufficiency of disclosure - (yes)
Inventive step - (yes)

Decisions cited:

Catchword:

Europäisches Patentamt
European Patent Office
Office européen des brevets

Beschwerdekammern

Boards of Appeal

Chambres de recours

Boards of Appeal of the
European Patent Office
Richard-Reitzner-Allee 8
85540 Haar
GERMANY
Tel. +49 (0)89 2399-0
Fax +49 (0)89 2399-4465

Case Number: T 0058/16 - 3.5.06

D E C I S I O N
of Technical Board of Appeal 3.5.06
of 29 June 2021

Appellant: (Applicant)	Fujitsu Limited 1-1, Kamikodanaka 4-chome Nakahara-ku Kawasaki-shi, Kanagawa 211-8588 (JP)
Representative:	Haseltine Lake Kempner LLP Cheapside House 138 Cheapside London EC2V 6BJ (GB)
Decision under appeal:	**Decision of the Examining Division of the European Patent Office posted on 19 June 2015 refusing European patent application No. 08790429.8 pursuant to Article 97(2) EPC.**

Composition of the Board:

Chairman	M. Müller
Members:	A. Teale
	B. Müller

Summary of Facts and Submissions

I. This is an appeal against the decision, dispatched with reasons on 19 June 2015, to refuse European patent application No. 08 790 429.8 on the basis that claim 1 of the main request did not comply with Article 123(2) EPC, regarding added subject-matter, and with Article 83 EPC, regarding sufficiency of disclosure. Claim 1 of the auxiliary request did not comply with Article 83 EPC, regarding sufficiency of disclosure.

II. On 1 February 2011 the application entered the European regional phase. The description pages intended for proceedings before the EPO were pages 1, 3 to 10 and 16 translated from international patent application WO 2010/016097 A1, which was published in Japanese, and amended pages 2 and 11 to 15, all submitted on entry into the European phase. The intended drawings were pages 1 to 10 translated from the published international patent application and page 11, all submitted on entry into the European phase.

III. A notice of appeal and the appeal fee were received on 27 July 2015. The appellant contested the decision in its entirety and requested that it be set aside. An auxiliary request was made for oral proceedings.

IV. With a statement of grounds of appeal, received on 29 October 2015, the appellant submitted claims according to a main and an auxiliary request and requested that the decision be set aside and a patent granted. The auxiliary request for oral proceedings was reiterated.

V. In an annex to a summons to oral proceedings the board summarized its preliminary opinion on the appeal as follows. The invention seemed to be sufficiently disclosed, Article 83 EPC. The application seemed to contain added subject-matter, Article 123(2) EPC, although not for the reasons given in the decision. The board had doubts regarding the clarity of the claims, Article 84 EPC, and the claimed subject-matter seemed not to involve an inventive step, Article 56 EPC, in view of the following document:

D1: GB 2 383 652 A.

VI. With a response received on 1 June 2021, the appellant filed amended claims according to a first and a second auxiliary request and stated that it would attend the oral proceedings.

VII. On the morning of the scheduled oral proceedings the chairman of the board telephoned the representative and indicated the board's preliminary opinion that it was still inclined not to allow the main request, but was inclined to remit the case with an order to grant on the basis of the first auxiliary request. The appellant then sent a telefax to the board stating that,

"[if] the board is prepared to remit the case to the 1st instance with an order to grant a patent on the basis of the 1st Auxiliary Request, the Main request is hereby withdrawn. Thus the 1st Auxiliary Request as filed on 1 June 2021 is now the (new) Main Request."

The oral proceedings were subsequently cancelled.

VIII. The application is being considered in the following form:

Description (both requests):
pages 2 to 16, received on 1 February 2011, and
pages 1 and 1a, received on 6 February 2012.

Claims:
Main request: 1 to 2, received with the letter of 1 June 2021 as "first auxiliary request".
Second auxiliary request: 1 to 2, received with the letter of 1 June 2021.

Drawings (both requests):
Pages 1/11 to 11/11, received on 1 February 2011.

IX. Claims according to the (new) main request, which are both independent, read a follows:

"1. An arithmetic processing unit (0) comprising:
an arithmetic section (3) for performing arithmetic processing;
a register file (1) having K number of register windows each of which has N number of registers, a current window selecting unit for selecting one of the K number of register windows as a current window according to window address information, and a register selecting unit for selecting a reading register from N number of registers included in the selected current window according to reading address information, the register windows each including a plurality of in-registers, a plurality of local registers, and a plurality of out-registers, the in-register included in a specified register window being shared with an out-register included in a register window before the specified register window, and an in-register included in the specified register window and the out-register included in the register window before the specified register

window being used for passing an argument when calling a subroutine;
a control section (4) for outputting window address information used by the current window selecting unit to select a current window,
wherein, when instructions in a program are described in an order of first instruction, SAVE instruction, and a second instruction other than the SAVE instruction, or in an order of first instruction, RESTORE instruction, and a second instruction other than the RESTORE instruction, and the second instruction is out-of-order executed, pipelines of the SAVE instruction and the RESTORE instruction each including Fetch (F), Decode (D), Dispatch (P), Buffer Read (B), Execute (X), Update Buffer (U), and Commit (W), the register selecting unit:
selects a local register whose window number is 2n, a local register whose window number is 2n+1, an out-register whose window number is 2n, an out-register whose window number is 2n-1, and an out-register whose window number is 2n+1 for an output of a first port, an output of a second port, an output of a third port, an output of a fourth port, and an output of a fifth port, respectively, during execution of any of Dispatch (P), Buffer Read (B), Execute (X), and Update Buffer (U) in the pipeline of the SAVE instruction executed after an execution start of the first instruction and a current window of the window address information indicates 2n (n indicating an integer),
selects a local register whose window number is 2n+2, the local register whose window number is 2n+1, the out-register whose window number is 2n, the out-register whose window number is 2n+1, and an out-register whose window number is 0 instead of 2n+2 for the output of the first port, the output of the second port, the output of the third port, the output of the

fourth port, and the output of the fifth port, respectively, during execution of any of Dispatch (P), Buffer Read (B), Execute (X), and Update Buffer (U) in the pipeline of the SAVE instruction executed after an execution start of the first instruction and a current window of the window address information indicates 2n+1,
selects the local register whose window number is 2n, a local register whose window number is 2n-1, the out-register whose window number is 2n, the out-register whose window number is 2n-1, and an out-register whose window number is 2n-2 for the output of the first port, the output of the second port, the output of the third port, the output of the fourth port, and the output of the fifth port, respectively,
during execution of any of Dispatch (P), Buffer Read (B), Execute (X), and Update Buffer (U) in the pipeline of the RESTORE instruction executed after an execution start of the first instruction and a current window of the window address information indicates 2n, and
selects the local register whose window number is 2n, the local register whose window number is 2n+1, the out-register whose window number is 2n, the out-register whose window number is 2n+1, and the out-register whose window number is 2n-2 for the output of the first port, the output of the second port, the output of the third port, the output of the fourth port, and the output of the fifth port, respectively,
during execution of any of Dispatch (P), Buffer Read (B), Execute (X), and Update Buffer (U) in the pipeline of the RESTORE instruction executed after an execution start of the first instruction and a current window of the window address information indicates 2n+1, and
a global port for selecting registers from global registers shared among all subroutines, and the first to fifth ports are connected to the register selecting

unit, a register value is selected from register values output from the global port and the first to fifth ports according to the reading address information and then is output.

2. A method of reading a register file (1), the register file having a plurality of registers, wherein the register file is divided into K number of register windows each of which has N number of registers, wherein the registers are read by:
a first step of selecting one of the K number of register windows as a current window according to window address information; and
a second step of selecting a reading register from N number of registers included in the selected current window according to reading address information, the register windows each including a plurality of in-registers, a plurality of local registers, and a plurality of out-registers, the in-register included in a specified register window being shared with an out-register included in a register window before the specified register window, and an in-register included in the specified register window and the out-register included in the register window before the specified register window being used for passing an argument when calling a subroutine,
wherein when instructions in a program are described in an order of first instruction, SAVE instruction, and a second instruction other than the SAVE instruction, or in an order of first instruction, RESTORE instruction, and a second instruction other than the RESTORE instruction, and the second instruction is out-of-order executed, pipelines of the SAVE instruction and the RESTORE instruction each including Fetch (F), Decode (D), Dispatch (P), Buffer Read (B), Execute (X), Update Buffer (U), and Commit (W), the method comprising:

selecting a local register whose window number is 2n, a local register whose window number is 2n+1, an out-register whose window number is 2n, an out-register whose window number is 2n-1, and an out-register whose window number is 2n+1 for an output of a first port, an output of a second port, an output of a third port, an output of a fourth port, and an output of a fifth port, respectively, during execution of any of Dispatch (P), Buffer Read (B), Execute (X), and Update Buffer (U) in the pipeline of the SAVE instruction executed after an execution start of the first instruction and a current window of the window address information indicates 2n (n indicating an integer),
selecting a local register whose window number is 2n+2, the local register whose window number is 2n+1, the out-register whose window number is 2n, the out-register whose window number is 2n+1, and an out-register whose window number is 0 instead of 2n+2 for the output of the first port, the output of the second port, the output of the third port, the output of the fourth port, and the output of the fifth port, respectively, during execution of any of Dispatch (P), Buffer Read (B), Execute (X), and Update Buffer (U) in the pipeline of the SAVE instruction executed after an execution start of the first instruction and a current window of the window address information indicates 2n+1,
selecting the local register whose window number is 2n, a local register whose window number is 2n-1, the out-register whose window number is 2n, the out-register whose window number is 2n-1, and an out-register whose window number is 2n-2 for the output of the first port, the output of the second port, the output of the third port, the output of the fourth port, and the output of the fifth port, respectively, during execution of any of Dispatch (P), Buffer Read (B), Execute (X), and

Update Buffer (U) in the pipeline of the RESTORE instruction executed after an execution start of the first instruction and a current window of the window address information indicates 2n, and
selecting the local register whose window number is 2n, the local register whose window number is 2n+1, the out-register whose window number is 2n, the out-register whose window number is 2n+1, and the out-register whose window number is 2n-2 for the output of the first port, the output of the second port, the output of the third port, the output of the fourth port, and the output of the fifth port, respectively, during execution of any of Dispatch (P), Buffer Read (B), Execute (X), and Update Buffer (U) in the pipeline of the RESTORE instruction executed after an execution start of the first instruction and a current window of the window address information indicates 2n+1,
and the method further comprising: selecting registers from global registers shared among all subroutines, selecting a register value from register values output from a global port and the first to fifth ports according to the reading address information, and outputting the selected register value."

Given the decision on the main request, the wording of the claims according to the second auxiliary request is immaterial to this decision.

Reasons for the Decision

1. The admissibility of the appeal

In view of, in particular, the facts set out at points I, III and IV above, the appeal fulfills the admissibility requirements under the EPC.

2. Summary of the invention

2.1 The application relates to a pipelined RISC (Reduced Instruction Set Computer) arithmetic processing unit. Known RISC processors carry out register-to-register operations and comprise a "master register file" (MRF) having many registers for temporarily storing the input data for and output data from arithmetic operations. The invention concerns high-speed reading from a register file without the need for a temporary memory.

2.2 As shown in figure 1 (see [3-8]), a register file (1000) comprises a plurality of overlapping "windows" W, each comprising an "in" segment, a "local" segment and an "out" segment, each segment comprising eight registers. The "out" segment is used for passing arguments to a subroutine called by the subroutine associated with that window, termed the "current window", whilst the "in" segment is used for receiving arguments from subroutines which have called that subroutine. As illustrated in figure 1, when a subroutine is called (termed "SAVE") the current window (W0) is "rotated" two places clockwise so that W1 becomes the current window. Likewise, when a subroutine is restored (termed "RESTORE"), the current window is rotated two places anti-clockwise so that W7 becomes the current window. The CWP (Current Window Pointer) stores the window number, 0 in this example, of the current window (W0). Incrementing or decrementing the CWP is termed "window switching". The file register also comprises a "global window" having eight global registers which store data used in all subroutines. As set out on page 3, lines 2 to 10, "overlapping" sets of registers (such as "W0 out" and "W1 in" in figure 1) are realised as a single set. In order for the processor to run a subroutine, it must be able to read data from, and write data to, the

registers in the MRF storing its input and output data, which limits the scale and speed of the circuitry for reading from the MRF.

2.3 It is known (see D1 below) to overcome these limitations using the circuit topology illustrated in figure 2 (see also figure 9 and [44-47]) which comprises a master register file (MRF 2001), a smaller working register file (WRF 2002), an arithmetic section (2003) and a control section (2004). To reduce the time taken to read register data and provide it to the arithmetic section, the WRF contains a copy (cache) of the window identified by the current window pointer (CWP) and, using read addresses provided by the control section, is read by the arithmetic section; see [10]. This configuration has hardware and power consumption costs due to caching a subset of the MRF in the WRF and transferring data between the two.

2.4 The invention provides an alternative solution to the problem, which is suitable *inter alia* for pipelined processors which, as the application puts it, execute instructions out of program order (termed "out-of-order") and for which the order in which register window switching instructions are consequently executed may vary. The corresponding configuration, illustrated in figure 3, avoids the need for a WRF when data is read into the arithmetic section (see [13, 15]), thus reducing processor hardware costs and power consumption. The corresponding configuration of the register windows is shown in figure 4. The Master Register File (MRF; 0) is controlled by a window selection signal ("WINDOW_ADDRESS") and a register selection signal ("READ_ADDRESS") from the control section (4), and the register selected at each reading port of the MRF is set to each port in advance so that

reading can occur out of program order. The steps executed out of order, namely "dispatch" (P), "buffer read" (B), Execute (X) and update buffer (U), are illustrated in figure 6 which shows an "instruction pipe". The pipe starts by fetching (F) and decoding (D) an instruction and ends in a "commit" (W) step. The control section (4) generates the "WINDOW_ADDRESS" and "READ_ADDRESS" signals based on the value of the CWP (Current Window Pointer).

2.5 As in figure 1, the output registers of a window, for instance "W0 out", are the same as the input registers, for instance "W1 in", of an adjoining window, meaning that only "local" and "out" registers are implemented, as shown in figure 5.

2.6 Figure 7 shows three instructions passing through the pipeline, the instructions (labelled using the nomenclature of figure 6) being an instruction with a current window pointer (CWP) of 1, a SAVE instruction (which is a switching instruction that increments the CWP from 1 to 2; see page 2, lines 25 to 29) and an instruction with a CWP of 2. Figure 7 shows three phases in which firstly only the CWP=1 instruction is executable, followed by a phase in which the instructions with CWP=1 and CWP=2 are executable and finally a phase in which only the CWP=2 instruction is executable. The net result is that the second instruction can be executed "out-of-order" before the window switch; see [49]. This requires that the register groups "W global", "W2 local", "W2 out", "W1 local", "W1 out" and "W0 out" be readable; see [27]. Likewise, when an instruction with CWP=1, a RESTORE instruction (which is a switching instruction that decrements the CWP from 1 to 0; see page 2, lines 25 to 29) and an instruction with a CWP=0 are executed

out of order then the registers "W global", "W1 local", "W1 out", "W0 local", "W0 out" and "W7 out" must be readable; see [28]. In both the "SAVE" and "RESTORE" cases, described above, five ports are required for reading the respective sets of registers from the MRF, the ports being designated "G" (global), "L" (local) and "OUT"; see figure 8. Six rules are disclosed for defining the register selections for the five MRF reading ports for even and odd CWPs except when SAVE and RESTORE are being executed (rules 1 and 2), even and odd CWPs when SAVE is being executed (rules 3 and 4) and even and odd CWPs when RESTORE is being executed (rules 5 and 6); see [34].

2.7 The independent claims 1 and 2 set out an arithmetic processing unit and a corresponding method, respectively, implementing rules 3 to 6 (see [34]) in the context of three instructions, namely a first instruction, either a SAVE or a RESTORE instruction, and a second instruction other than the SAVE or RESTORE instruction, respectively, which is out-of-order executed. The expression "current window selecting unit" has been amended to "current register selecting unit".

3. The board's understanding of the invention

The invention centres on being able, under certain circumstances, to execute an additional pipelined instruction out-of-order before a window switch due to a SAVE/subroutine call or RESTORE/subroutine return. This is achieved by providing the arithmetic processing unit access to more register sets of the master read file (MRF) (see figures 8 to 10) than would otherwise be required by means of, although it is not expressly referred to as such, essentially a multiplexing arrangement between the MRF and the arithmetic

processing unit implementing rules 3 to 6, set out in the claims and the description; see pages 10 and 11.

4. Clarity and support, Article 84 EPC

4.1 In the annex to the summons the board raised the objection that the expression in claim 1 of the previous main request "when an instruction in a program is described in an order of first instruction, SAVE instruction, and a second instruction other than the SAVE instruction" was unclear, contrary to Rule 49(11) EPC, in that it seemed to use the term "instruction" with two different meanings, namely a group of instructions and an individual instruction. The same objection applied *mutatis mutandis* to claim 2. Claims 1 and 2 have now been amended in a way which overcomes this objection, stating that "wherein, when instructions in a program are described in an order of first instruction, SAVE instruction, and a second instruction other than the SAVE instruction ..." Thus it is now clear that an "instruction" is an individual instruction.

4.2 In the annex to the summons the board objected that claims 1 and 2 of the previous main request used the expression "window**s** address information" (emphasis by the board) in numerous places, except in claim 1, page 17, lines 25 to 26, which used the different expression "window address information", apparently with the same meaning. The use of two different expressions with the same meaning was contrary to Rule 49(11) EPC and made claim 1 unclear, Article 84 EPC. The claims have now been amended to consistently use the expression "window address information" throughout, thus overcoming the objection.

4.3 Regarding the features corresponding to rule 4 (see pages 10-11), set out in claims 1 and 2 of the previous main request on page 18, lines 15 to 26, and page 20, line 27, to page 21, line 3, respectively, the board expressed doubts in the annex to the summons whether the expression "and an out-register whose window number is 0 instead of 2n+2 for the output of the first port" (see lines 18-19 of claim 1 and lines 30-31 of claim 2) was supported by the description. The appellant has pointed to the disclosure on page 10, lines 10 to 13, as providing support for this feature. The board agrees that this passage does indeed provide support for the above feature, Article 84 EPC.

5. Added subject-matter, Article 123(2) EPC

5.1 According to the appealed decision (point 12), the expression in claim 1 of the main request "the current window selecting unit selects a local register" was not disclosed in the application as originally filed. According to original paragraph 20 and figure 3, the current window selecting unit selected the number of the current window, while selection of a respective register within that particular register window was performed by the register selecting unit. The decision did not raise the same objection against claim 1 of the then auxiliary request in which the expression "current window" had been replaced by "register".

5.2 Present claim 1 has been amended along the lines of the auxiliary request in the decision, the amendment being based on page 9, line 25, to page 11, line 29, of the original application. As the expression in claim 1 "the current window selecting unit" has been amended to "the register selecting unit", the board agrees with the appellant that the objection has been overcome, since

the application repeatedly states that the register selecting unit selects registers, as its name implies; see, for instance, original claim 1.

6. Sufficiency of disclosure, Article 83 EPC

6.1 According to the appealed decision (points 13 to 13.5), claim 1 taught the selection of five different registers, depending on whether a window changing instruction was currently being executed in the processor's pipeline, on the instruction type (SAVE/ RESTORE) and on the current window information. However, the skilled person was not taught which particular instruction window should be accessed when instructions were executed out of order (see 13.1). The application also did not provide any information about how to define an instruction's position in the program order in relation to a SAVE or a RESTORE instruction. The appellant has responded that rules 3 to 6, set out in claim 1, define the selection of register windows during out-of-order instruction execution. Although the invention did not define the second instruction's position in program order, the description set out the circumstances (the order of the relevant instructions) under which out-of-order execution was possible. The board agrees with the appellant on this point. Given the explanation of the circumstances under which out-of-order execution is possible and its implementation (see figure 8 and [26-28, 32-34]) in the description and claim 1 (page 17, lines 28 to 33, and page 18, line 2, to page 19, line 13), the person skilled in the art of arithmetic processing unit design would be able to identify opportunities for out-of-order execution and to implement them.

6.2 Returning to the decision (point 13.2), claim 1 set out the selection of five different registers from different register sub-groups, and the skilled person would not have known how to execute an arithmetic instruction which read two operands from a local register window and wrote the result to a third register in the same local register window. The appellant has argued that the application (see figures 2; see WRITE_ADDRESS and 10; see WRITE_DATA) discloses the writing of results to a local register window. The board notes that claims 1 and 2 do not specify any out-of-order execution so that, in the board's judgment, no sufficiency objection follows from omitting to specify this out-of-order execution in more detail. Moreover the invention relates to **reading** data more quickly from the MRF (see "reading port" in [15, 21, 33, 48]), the independent claims setting out a "reading register" and "reading address information", so that it need not disclose further details of **writing** results.

6.3 According to the decision (points 13.3 to 13.5), the application failed to teach what would happen if a program sequence of a first instruction, a SAVE instruction and a RESTORE instruction or a program sequence of a first instruction, a RESTORE instruction and a SAVE instruction were executed. It was clear that a controlling mechanism was necessary to determine whether a respective second instruction could be executed out-of-order in respect of a window switching instruction. However, such a mechanism was not disclosed in the application. The same controlling mechanism was needed to prevent a second instruction from executing out-of-order when the first instruction updated a register that would subsequently be read by the second instruction; see point 13.4. Such a mechanism was also not disclosed by the application,

which only taught (see paragraph 22) that a WRF (working register file) was not necessary and did not explain how the register file was updated. Furthermore, the application did not explain what would happen if a program sequence of a first instruction, a SAVE instructions, a second instruction, a SAVE instruction and a third instruction were to be executed by the processor's pipeline; see point 13.5. The appellant has argued that the skilled person would realize that a switching instruction (SAVE/RESTORE) could not be executed out-of-turn and could use the teaching of the application to analyse the above scenarios into units which were dealt with in the description. The board takes the view that the subject-matter of claim 1 implies the controlling functions mentioned in the decision, so that the skilled person could implement them without undue burden. Rule 42(1)(e) EPC requires that the application "describe in detail at least one way of carrying out the invention claimed, using examples where appropriate and referring to the drawings, if any". Thus the EPC does not require that an application deal with every conceivable "what if"-scenario. The board also emphasises that the claims do not specify exactly how individual instructions are "out-of-order executed", although it is plausible from the description that the invention can be used for that purpose. This point is - however ultimately not crucial for inventive step (see point 8 below).

6.4 Hence the board finds that the invention is sufficiently disclosed in the application, Article 83 EPC.

7. Document D1 (GB 2 383 652 A)

7.1 D1 relates to an information processing apparatus having a master register file (MRF) and a working

register file (WRF) which holds data of the register windows of the current window and of the preceding and following windows; see abstract.

7.2 As stated in the European Search Opinion (points 2-2.3), D1 discloses an arithmetic processing unit comprising an arithmetic section for performing arithmetic processing (figure 14; 12), a register file having K windows, each of which has N registers including a shared register that is shared by a neighboring window, and is used for passing an argument when calling a subroutine (figure 14; 10), a current window selecting unit for selecting one of the K windows, and a register selecting unit for selecting a reading register via a reading address (figure 21; see page 39, line 25, to page 40, line 15) and a control section for outputting a window address signal for the current window selecting unit of the register file to select a window (figure 21).

7.3 The board notes that D1 further relates to a RISC architecture (see page 1, lines 13 to 19) and discloses a cyclical register window structure in figures 1 and 5 very similar to that in figure 1 of the present application; see also page 2, lines 9 to 25, regarding global registers, shared registers and incrementing (SAVE)/decrementing (RESTORE) the current window pointer (CWP). D1 concerns the performance improvements accruing from a WRF storing data from three windows (see figure 6), rather than the prior art case of the WRF only storing data relating to a single window; see figure 2, page 4, lines 9 to 15, and page 14, lines 1 to 9.

7.4 The board notes that D1 also mentions in its abstract (last sentence) executing instructions out-of-order,

even if a SAVE or RESTORE instruction is successively executed, thus improving process efficiency. Out of order execution is explained on page 1, lines 5 to 12, and page 12, lines 4 to 15, as meaning that an instruction goes ahead of a window switch. The board understands this to mean that, instead of the execution order <instruction 1><window switch><instruction 2>, the order becomes <instruction 1><instruction 2><window switch>.

7.5 Hence it is common ground between the board and the appellant (see the applicant's response of 3 February 2015, page 4, first three paragraphs) that D1 does not disclose the selection of registers according to rules 3 to 6 for five ports, set out in the claims of both requests.

7.6 In the same response the appellant argued that D1 also did not disclose the advantageous effects of being able to read data into the arithmetic section at high speed without having the WRF as a temporary memory, and being able to execute an instruction subsequent to a window switching instruction out of order. In view of the above summary of D1, the board disagrees regarding the execution of an instruction subsequent to a window switching instruction out-of-order; this is indeed known from D1.

8. Inventive step, Article 56 EPC

8.1 Regarding the obviousness of the difference features over D1, as the board stated in the annex to the summons, there is no obvious technical reason for the skilled person starting from D1 to dispense with the WRF, since D1 teaches the performance advantages accruing from enlarging the WRF to store data regarding

not only one but three consecutive windows (see, for example, page 8, lines 1 to 10). Even if the WRF were to be removed (or if the inventive step assessment were to start from the earlier embodiment that does not comprise a WRF; see page 3, lines 15 to 20), the board also sees no obvious reason for the skilled person to implement the multiplexing arrangement according to rules 3 to 6, set out in the claims, to read data from the five ports of the MRF.

8.2 Turning to the technical effect of the multiplexing arrangement according to rules 3 to 6, set out in the claims, the board in the annex pointed out that "out-of-order" execution of instructions was already known from D1. In the response of 1 June 2021 the appellant argued that the multiplexing arrangement according to rules 3 to 6 had the effect of avoiding the complex circuitry of the WRF and the need to move data between the MRF and the WRF. Hence the claimed subject-matter set out an alternative to the use of a cache memory (WRF) known from D1 which was simpler and required less clock cycles to read register data. The board agrees with the appellant that these are technical effects which cause the difference features over D1 to contribute to inventive step, Article 56 EPC.

8.3 Consequently the board finds that the subject-matter of apparatus claim 1 involves an inventive step, Article 56 EPC, in view of D1. The same arguments apply *mutatis mutandis* to corresponding method claim 2.

Order

For these reasons it is decided that:

1. The decision under appeal is set aside.
2. The case is remitted to the first instance with the order to grant a patent in the following version:

Description: pages 1 and 1a, received on 6 February 2012, amended pages 2 and 11 to 15, received on 1 February 2011, and original pages 3 to 10 and 16, received on 1 February 2011.
Claims: 1 to 2 of the first auxiliary request, received on 1 June 2021.
Drawings: original drawing sheets 1/11 to 10/11, received on 1 February 2011, and amended drawing sheet 11/11, received on 1 February 2011.

The Registrar: The Chairman:

L. Stridde M. Müller

Decision electronically authenticated