





MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK,  
SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ,  
GW, KM, ML, MR, NE, SN, TD, TG).

— *as to the applicant's entitlement to claim the priority of  
the earlier application (Rule 4.17(iii))*

**Declarations under Rule 4.17:**

— *as to applicant's entitlement to apply for and be granted  
a patent (Rule 4.17(ii))*

**Published:**

— *with international search report (Art. 21(3))*

## METHOD AND APPARATUS FOR FORWARDING LITERAL GENERATED DATA TO DEPENDENT INSTRUCTIONS MORE EFFICIENTLY USING A CONSTANT CACHE

### Field of Disclosure

[0001] The invention relates to microprocessors.

### Background

[0002] In a typical central processing unit (CPU) pipeline flow, an instruction in the pipeline will first obtain its operands and then execute before finally writing back the result and possibly forwarding the result to subsequent dependent consuming instructions. Depending on the CPU microarchitecture, this process often occurs across multiple pipeline stages so as to optimize performance and frequency.

[0003] In a superscalar processor containing multiple execution pipelines, forwarding the result of one instruction to one or more consuming instructions in the pipelines may be a performance critical function that if not done efficiently may lead to pipeline stalls. A data dependency stall is the most common stall involving instructions attempting to dispatch to their respective pipelines for execution, where a stalled instruction waits for the producer of an operand to complete. Delays in forwarding the needed operand from its producer to the stalled instruction results in degraded CPU performance.

### SUMMARY

[0004] Embodiments of the invention are directed to systems and methods for forwarding literal generated data to dependent instructions more efficiently using a cache for storing constants (literals or immediates).

[0005] In an embodiment, a processor includes a register, a first pipeline, a cache, and a controller. The controller stores a value in an entry in the cache in response to the first pipeline decoding an instruction, wherein the instruction writes the value to the register upon completing execution, and wherein the value is determined or available when the first pipeline decodes the instruction. The controller sets a tag field in the entry to tag the entry with the register, and sets a flag field in the entry to indicate that the entry is valid. The instruction may be a move immediate instruction.

[0006] In another embodiment, a method includes decoding a first instruction in a first pipeline, wherein the first instruction writes a value to a register upon completing

execution, and wherein the value is determined or available when the first pipeline decodes the first instruction. The method further includes storing the value in an entry in a cache; tagging the entry with the register; and setting the entry as valid.

[0007] In another embodiment, a processor includes a first pipeline to decode a first instruction, wherein the first instruction writes a value to a register upon completing execution, and wherein the value is determined or available when the first pipeline decodes the first instruction. The processor further includes a means for storing, the means for storing to store the value in an entry in a cache; a means for tagging, the means for tagging to tag the entry with the register; and a means for setting, the means for setting to set the entry as valid.

[0008] In another embodiment, a non-transitory computer readable medium has stored instructions to cause a processor to perform a process. The process includes decoding a first instruction in a first pipeline, wherein the first instruction writes a value to a register upon completing execution, and wherein the value is determined or available when the first pipeline decodes the first instruction; storing the value in an entry in a cache; tagging the entry with the register; and setting the entry as valid.

### **BRIEF DESCRIPTION OF THE DRAWINGS**

[0009] The accompanying drawings are presented to aid in the description of embodiments of the invention and are provided solely for illustration of the embodiments and not limitation thereof.

[0010] Figure 1 illustrates a processor according to an embodiment.

[0011] Figure 2 illustrates a method according to an embodiment.

[0012] Figure 3 illustrates a wireless communication system in which embodiments may find application.

### **DETAILED DESCRIPTION**

[0013] Aspects of the invention are disclosed in the following description and related drawings directed to specific embodiments of the invention. Alternate embodiments may be devised without departing from the scope of the invention. Additionally, well-known elements of the invention will not be described in detail or will be omitted so as not to obscure the relevant details of the invention.

[0014] The word “exemplary” is used herein to mean “serving as an example, instance, or illustration.” Any embodiment described herein as “exemplary” is not necessarily to be construed as preferred or advantageous over other embodiments. Likewise, the term “embodiments of the invention” does not require that all embodiments of the invention include the discussed feature, advantage or mode of operation.

[0015] The terminology used herein is for the purpose of describing particular embodiments only and is not intended to be limiting of embodiments of the invention. As used herein, the singular forms “a”, “an” and “the” are intended to include the plural forms as well, unless the context clearly indicates otherwise. It will be further understood that the terms “comprises”, “comprising”, “includes” and/or “including”, when used herein, specify the presence of stated features, integers, steps, operations, elements, and/or components, but do not preclude the presence or addition of one or more other features, integers, steps, operations, elements, components, and/or groups thereof.

[0016] Further, many embodiments are described in terms of sequences of actions to be performed by, for example, elements of a computing device. It will be recognized that various actions described herein can be performed by specific circuits (e.g., application specific integrated circuits (ASICs)), by program instructions being executed by one or more processors, or by a combination of both. Additionally, these sequence of actions described herein can be considered to be embodied entirely within any form of computer readable storage medium having stored therein a corresponding set of computer instructions that upon execution would cause an associated processor to perform the functionality described herein. Thus, the various aspects of the invention may be embodied in a number of different forms, all of which have been contemplated to be within the scope of the claimed subject matter. In addition, for each of the embodiments described herein, the corresponding form of any such embodiments may be described herein as, for example, “logic configured to” perform the described action.

[0017] Figure 1 illustrates components of a processor 100, where for ease of illustration not all components are illustrated. Many processors are superscalar processor, employing more than one pipeline. Two pipelines are illustrated in Figure 1, labeled 102a and 102b, although in practice there may be more than two pipelines in a superscalar processor. For simplicity, three stages are shown in each pipeline, but in practice more than three stages are likely used.

[0018] Illustrated in the pipelines of Figure 1 are instruction fetch stages 104a and 104b, decode stages 106a and 106b, and execution stages 108a and 108b. A pipeline may include other stages such as register fetch, hazard checking, cache hit detection, data fetch, and write back for loads and register-to-register operations, to name a few examples. A controller functional unit, labeled 110, controls the pipelines 102a and 102b.

[0019] A move instruction is a commonly used instruction for moving (copying or writing) data from one location to another. A move instruction is often written as MOV, and that convention will be followed here. A common use of a move instruction is to copy the value of a constant into an architected register. The constant value to be copied may be referred to as an immediate or literal. A move instruction for moving a constant to a register may be termed a move immediate instruction and written as MOV *Rm* #*constant*, where *constant* refers to the constant value and *Rm* refers to the architected register to which the constant value is written. In Figure 1, the register *Rm* is labeled 118 and is illustrated as a register within the register file 120.

[0020] Upon decoding a move immediate instruction, an embodiment stores the constant as part of an entry in a cache, referred to as a constant cache and labeled 112 in Figure 1. An entry in the constant cache 112 is labeled 114 in Figure 1, and comprises three fields: a tag field labeled 114a, a constant field labeled 114b, and a flag field labeled 114c. As its name implies, the constant field 114b stores the constant value associated with the entry. The tag field 114a identifies the register to which the constant value is to be written (or moved). The flag field 114c comprises one or more bits to indicate the status of the entry 114. For some embodiment, the flag field 114c may be one bit in width, indicating whether the entry is valid or not.

[0021] The constant cache 112 may be realized in the processor 100 as a register file. In the illustration of Figure 1, the constant cache 112 is shown as a separate structure from the register file 120. However, the constant cache 112 need not necessarily be independent of the register file 120. For example, the constant cache 112 may be part of the register file 120, or both structures may be included in a larger register file structure.

[0022] A move immediate instruction requires no subsequent execution to calculate its result. Typically, when a constant is generated, it is consumed immediately by a subsequent (in program order) consuming instruction. By utilizing the constant cache

112, subsequent consuming instructions have access to the stored constant value before the constant value is written to the destination architected register.

[0023] The contents of the constant cache 112 may be viewed as being organized into a table, where the constant value stored in an entry is written by a move immediate instruction and tagged according to the destination register of the move immediate instruction. Consider a result (the constant value) of a move immediate instruction stored in the constant cache 112 and a subsequent (in program order) instruction that depends upon the move immediate instruction, where an operand of the subsequent instruction is the constant value that the move immediate instruction is to move to a destination register. The subsequent instruction is the consuming instruction, and the destination register is the register targeted by the move instruction.

[0024] For an embodiment, execution of the consuming instruction need not wait for the result of the move immediate instruction to be forwarded, nor wait for the move immediate instruction to complete execution. Rather, the consuming instruction may use as its operand the constant value stored in the entry in the constant cache 112 associated with the move immediate instruction that it depends upon. As a result, no data forwarding is required and no data stall need occur regardless of whether the move immediate instruction has completed or is still in a pipeline.

[0025] Furthermore, the move immediate instruction and the data dependent consuming instruction may be at the same stage in different pipelines, and yet for some embodiments the data dependent consuming instruction may obtain its operand with zero pipeline cycle delay.

[0026] When a move immediate instruction is decoded and its immediate (literal or constant value) is stored in an entry in the constant cache 114, the flag field 114c associated with the entry is set to indicate that the contents of the entry are valid. When that entry is later accessed by a consuming instruction, the validity of an entry is checked before the immediate stored in the entry is forwarded to the consuming instruction. If the flag field associated with an entry indicates that the immediate stored in the entry is not valid, then the stored immediate is not forwarded to the consuming instruction.

[0027] Although the above description is within the context of a move immediate instruction, embodiments are not limited to move immediate instructions when employing the constant cache 112. The controller 110 may be configured so that for

other types of instructions that write values to a destination register, an entry may be generated in the constant cache 112 as described with respect to the move immediate instruction, so that the stored value may be forwarded to a consuming instruction. Examples of such instructions are branch and link instructions, and program control relative branches, to name a few.

[0028] More generally, the described embodiments may be apply to instructions that write a result to the register file, where the result can be determined by either information contained in the decode of the instruction or available at the time of decode. Such instructions do not have any operands that must read the register file. However, for ease of discussion, the embodiments disclosed herein are described for a move immediate instruction, where a move immediate instruction merely serves as example instruction for which embodiments may be of utility.

[0029] When an instruction writes a result to an architected register, where the instruction needs to read from the register file before execution to determine the result, then the controller 110 invalidates any entry in the constant cache 114 with a tag matching the architected register. In this case, the controller 110 sets the flag field of the matching entry to a value indicating that the constant value stored in the entry is not valid.

[0030] Controller 110 updates entries in the constant cache 114 according to the above-described embodiments. These actions are may be performed completely by hardware. For some embodiments, instructions stored in a memory, such as for example the memory 116, may carry out the above-described actions. The memory 116 may in general be a non-transitory computer readable medium.

[0031] Figure 2 illustrates the above-described actions. In step 202 an instruction is decoded. In step 204 the decoded instructions is a move immediate instruction, denoted as  $\text{MOV } R_m \#C$  to indicate that a constant value  $C$  is to be moved into architected register  $R_m$ . Upon decoding the move immediate instruction, step 206 indicates that the constant value  $C$  is stored in an entry in the constant cache 112, where the entry is tagged with the register  $R_m$ , and the flag field of the entry is set to indicate that the entry is valid.

[0032] If the decoded instruction is a consumer of the architected register  $R_m$ , as indicated in step 208, then provided there is a valid entry in the constant cache 112 associated (tagged) with the architected register  $R_m$ , the constant value  $C$  stored in the



constant field of that entry is forwarded to the consumer, as indicated in step 210. If the decoded instruction is an instruction that completes execution and writes (or copies) a constant value to the architected register  $R_m$ , as indicated in step 212, then the controller 110 invalidates the entry (provided there is one) in the constant cache 112 associated (tagged) with the architected register  $R_m$ , as indicated in step 214.

[0033] Figure 3 illustrates a wireless communication system in which embodiments may find application. Figure 3 illustrates a communication network 302 comprising base stations 304A, 304B, and 304C. Figure 3 shows a communication device, labeled 306, which may be a mobile cellular communication device such as a cellular phone (e.g., a smart phone), a tablet, or other kind of communication device suitable for a cellular phone network, such as a computer system. The communication device 306 need not be mobile. In the particular example of Figure 3, the communication device 306 is located within the cell associated with the base station 304C. Arrows 308 and 310 pictorially represent the uplink channel and the downlink channel, respectively, by which the communication device 306 communicates with the base station 304C.

[0034] Embodiments may be used in data processing systems associated with the communication device 306, or with the base station 304C, or both, for example. Figure 3 illustrates only one application among many in which the embodiments described herein may be employed.

[0035] Those of skill in the art will appreciate that information and signals may be represented using any of a variety of different technologies and techniques. For example, data, instructions, commands, information, signals, bits, symbols, and chips that may be referenced throughout the above description may be represented by voltages, currents, electromagnetic waves, magnetic fields or particles, optical fields or particles, or any combination thereof.

[0036] Further, those of skill in the art will appreciate that the various illustrative logical blocks, modules, circuits, and algorithm steps described in connection with the embodiments disclosed herein may be implemented as electronic hardware, computer software, or combinations of both. To clearly illustrate this interchangeability of hardware and software, various illustrative components, blocks, modules, circuits, and steps have been described above generally in terms of their functionality. Whether such functionality is implemented as hardware or software depends upon the particular application and design constraints imposed on the overall system. Skilled artisans may

implement the described functionality in varying ways for each particular application, but such implementation decisions should not be interpreted as causing a departure from the scope of the present invention.

[0037] The methods, sequences and/or algorithms described in connection with the embodiments disclosed herein may be embodied directly in hardware, in a software module executed by a processor, or in a combination of the two. A software module may reside in RAM memory, flash memory, ROM memory, EPROM memory, EEPROM memory, registers, hard disk, a removable disk, a CD-ROM, or any other form of storage medium known in the art. An exemplary storage medium is coupled to the processor such that the processor can read information from, and write information to, the storage medium. In the alternative, the storage medium may be integral to the processor.

[0038] Accordingly, an embodiment of the invention can include a computer readable media embodying a method for forwarding literal generated data to dependent instructions more efficiently using a constant cache.

[0039] Accordingly, the invention is not limited to illustrated examples and any means for performing the functionality described herein are included in embodiments of the invention.

[0040] While the foregoing disclosure shows illustrative embodiments of the invention, it should be noted that various changes and modifications could be made herein without departing from the scope of the invention as defined by the appended claims. The functions, steps and/or actions of the method claims in accordance with the embodiments of the invention described herein need not be performed in any particular order. Furthermore, although elements of the invention may be described or claimed in the singular, the plural is contemplated unless limitation to the singular is explicitly stated.

## CLAIMS

### WHAT IS CLAIMED IS:

1. An apparatus comprising:
  - a register;
  - a first pipeline;
  - a cache; and
  - a controller to store a value in an entry in the cache in response to the first pipeline decoding an instruction, wherein the instruction writes the value to the register upon completing execution, and wherein the value is determined or available when the first pipeline decodes the instruction;
  - the controller to set a tag field in the entry to tag the entry with the register, and to set a flag field in the entry to indicate the entry is valid.
2. The apparatus of claim 1, wherein the instruction is a move immediate instruction.
3. The apparatus of claim 1, further comprising a register file, the register file comprising the register, the controller to set the flag field in the entry to indicate the entry is invalid upon the first pipeline decoding a second instruction targeting the register, the second instruction determining its result by reading from the register file.
4. The apparatus of claim 1, the controller, in response to the first pipeline decoding a consuming instruction subsequent in program order to the instruction and having an operand naming the register, to
  - search the cache for the entry tagged with the register; and
  - forward the value to the first pipeline provided the entry is found and provided the flag field of the entry indicates the entry is valid.
5. The apparatus of claim 4, further comprising a register file, the register file comprising the register, the controller to set the flag field in the entry to indicate the entry is invalid upon the first pipeline decoding a second instruction in a decode stage,

the second instruction targeting the register and determining its result in a pipeline stage subsequent to the decode stage by reading from the register file.

6. The apparatus of claim 1, further comprising:

a second pipeline;

the controller to forward to the second pipeline the value stored in the entry tagged with the register upon the second pipeline decoding a consuming instruction, the consuming instruction subsequent in program order to the instruction and having the register as an operand, provided the flag field of the entry indicates the entry is valid.

7. The apparatus of claim 1, further comprising:

a second pipeline, wherein the first and second pipelines each comprise respective decode stages,

the controller to forward to the second pipeline the value when the instruction is in the decode stage of the first pipeline and the consuming instruction is in the decode stage of the second pipeline, provided the instruction is to cause the controller to write the flag field of the entry as valid.

8. The apparatus of claim 1, wherein the apparatus is selected from the group consisting of a cellular phone and a base station.

9. A method comprising:

decoding a first instruction in a first pipeline, wherein the first instruction writes a value to a register upon completing execution, and wherein the value is determined or available when the first pipeline decodes the first instruction;

storing the value in an entry in a cache;

tagging the entry with the register; and

setting the entry as valid.

10. The method of claim 9, wherein the first instruction is a move immediate instruction.

11. The method of claim 9, further comprising:

decoding a second instruction in the first pipeline, the second instruction subsequent in program order to the first instruction and a consuming instruction of the value; and

forwarding the value from the entry in the cache to the first pipeline as an operand for the second instruction, provided the entry is indicated valid.

12. The method of claim 11, further comprising:

decoding a third instruction in the first pipeline, the third instruction targeting the register, the third instruction determining its result by reading from a register file; and

setting the entry as invalid upon decoding the third instruction.

13. The method of claim 9, further comprising:

decoding a second instruction in the first pipeline, the second instruction targeting the register, the second instruction determining its result by reading from a register file; and

setting the entry as invalid upon decoding the second instruction.

14. The method of claim 9, further comprising:

decoding a second instruction in a second pipeline, the second instruction subsequent in program order to the first instruction and a consuming instruction of the value; and

forwarding the value from the entry in the cache to the first pipeline as an operand for the second instruction, provided the entry is indicated valid.

15. The method of claim 14, further comprising:

decoding a third instruction in the first pipeline, the third instruction targeting the register, the third instruction determining its result by reading from a register file; and

setting the entry as invalid upon decoding the third instruction.

16. The method of claim 9, further comprising:

decoding a second instruction in a second pipeline, the second instruction subsequent in program order to the first instruction and a consuming instruction of the value; and

forwarding the value from the first pipeline to the second pipeline as an operand for the second instruction with zero pipeline cycle delay, provided the first instruction causes the entry to be indicated valid when the first instruction executes.

17. An apparatus comprising:

a register;

a first pipeline to decode a first instruction, wherein the first instruction writes a value to the register upon completing execution, and wherein the value is determined or available when the first pipeline decodes the first instruction;

a means for storing, the means for storing to store the value in an entry;

a means for tagging, the means for tagging to tag the entry with the register;

and

a means for setting, the means for setting to set the entry as valid.

18. The apparatus of claim 17, wherein the first instruction is a move immediate instruction.

19. The apparatus of claim 17, further comprising:

a means for forwarding, the means for forwarding to forward the value from the entry to the first pipeline as an operand for a second instruction decoded in the first pipeline, provided the entry is indicated valid, the second instruction subsequent in program order to the first instruction and a consuming instruction of the value.

20. The apparatus of claim 19, further comprising a register file, the register file comprising the register, wherein the means for setting sets the entry as invalid upon the first pipeline decoding a third instruction targeting the register, the third instruction determining its result by reading from the register file.

21. The apparatus of claim 17, further comprising a register file, the register file comprising the register, wherein the means for setting sets the entry as invalid upon the

first pipeline decoding a second instruction targeting the register, the second instruction determining its result by reading from the register file.

22. The apparatus of claim 17, further comprising:

a second pipeline; and

a means for forwarding, the means for forwarding to forward the value from the entry to the second pipeline as an operand for a second instruction decoded in the second pipeline, the second instruction subsequent in program order to the first instruction and a consuming instruction of the value, provided the entry is indicated valid.

23. The apparatus of claim 22, further comprising a register file, the register file comprising the register, wherein the means for setting sets the entry as invalid upon the first pipeline decoding a third instruction, the third instruction targeting the register, the third instruction determining its result by reading from the register file.

24. The apparatus of claim 17, further comprising:

a second pipeline; and

a means for forwarding, the means for forwarding to forward the value to the second pipeline as an operand for a second instruction decoded in the second pipeline with zero pipeline cycle delay, the second instruction subsequent in program order to the first instruction and a consuming instruction of the value, provided the first instruction causes the entry to be indicated valid when the first instruction executes.

25. The apparatus of claim 17, wherein the apparatus is selected from the group consisting of a cellular phone and a base station.

26. A non-transitory computer-readable medium having stored instructions to cause a processor to perform a process comprising:

decoding a first instruction in a first pipeline, wherein the first instruction writes a value to a register upon completing execution, and wherein the value is determined or available when the first pipeline decodes the first instruction;

storing the value in an entry in a cache;

tagging the entry with the register; and  
setting the entry as valid.

27. The non-transitory computer-readable medium of claim 26, wherein the first instruction is a move immediate instruction.

28. The non-transitory computer-readable medium of claim 26, the process further comprising:

decoding a second instruction in the first pipeline, the second instruction subsequent in program order to the first instruction and a consuming instruction of the value; and

forwarding the value from the entry in the cache to the first pipeline as an operand for the second instruction, provided the entry is indicated valid.

29. The non-transitory computer-readable medium of claim 28, the process further comprising:

decoding a third instruction in the first pipeline, the third instruction targeting the register, the third instruction determining its result by reading from a register file; and

setting the entry as invalid upon decoding the third instruction.

30. The non-transitory computer-readable medium of claim 26, the process further comprising:

decoding a second instruction in the first pipeline, the second instruction targeting the register, the second instruction determining its result by reading from a register file; and

setting the entry as invalid upon decoding the second instruction.

31. The non-transitory computer-readable medium of claim 26, the process further comprising:

decoding a second instruction in a second pipeline, the second instruction subsequent in program order to the first instruction and a consuming instruction of the value; and



forwarding the value from the entry in the cache to the second pipeline as an operand for the second instruction, provided the entry is indicated valid.

32. The non-transitory computer-readable medium of claim 31, the process further comprising:

decoding a third instruction in the first pipeline, the third instruction targeting the register, the third instruction determining its result by reading from a register file; and

setting the entry as invalid upon decoding the third instruction.

33. The non-transitory computer-readable medium of claim 26, the process further comprising:

decoding a second instruction in a second pipeline, the second instruction subsequent in program order to the first instruction and a consuming instruction of the value; and

forwarding the value from the first pipeline to the second pipeline as an operand for the second instruction with zero pipeline cycle delay, provided the first instruction causes the entry to be indicated valid when the first instruction executes.

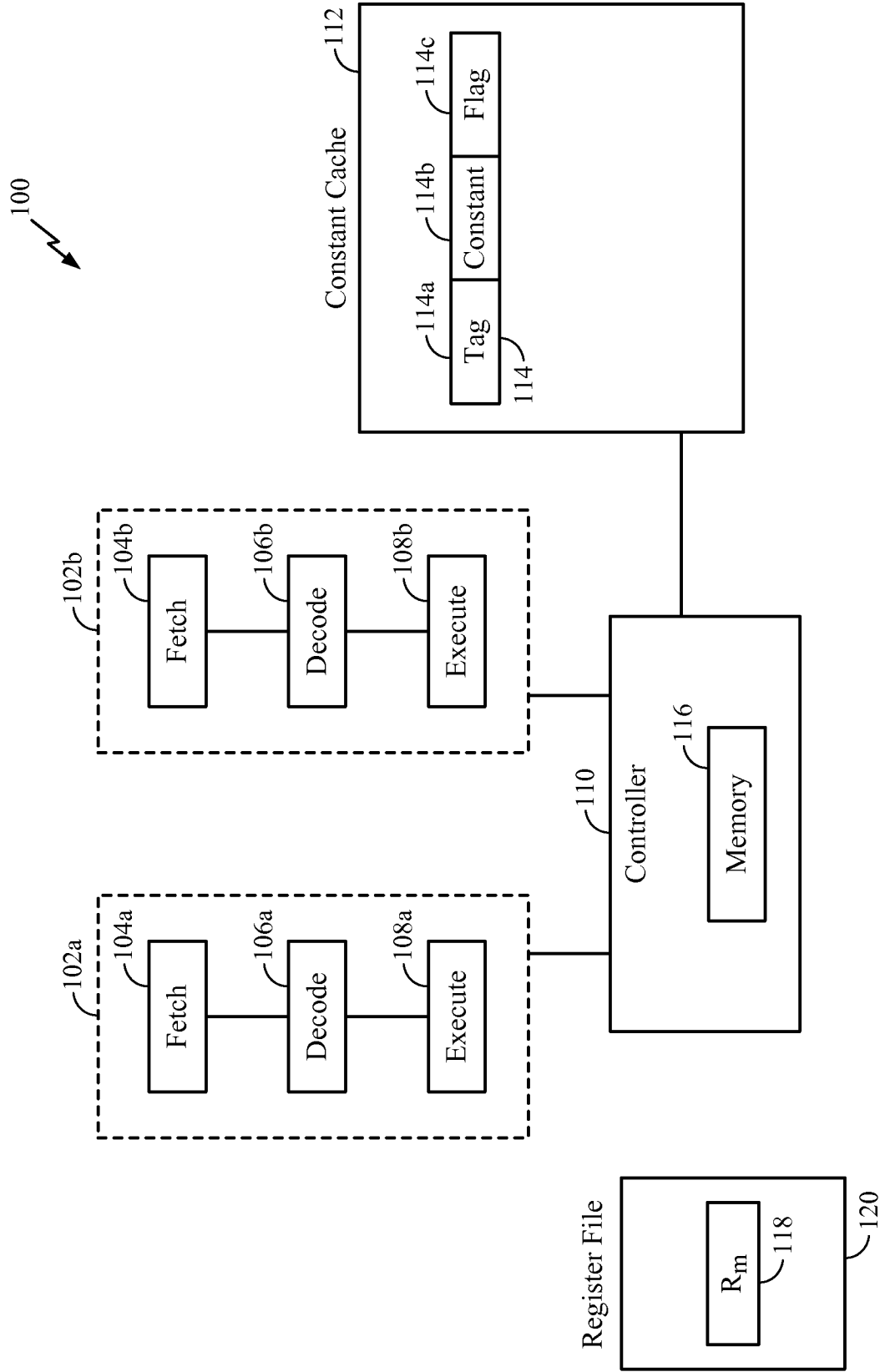


FIG. 1

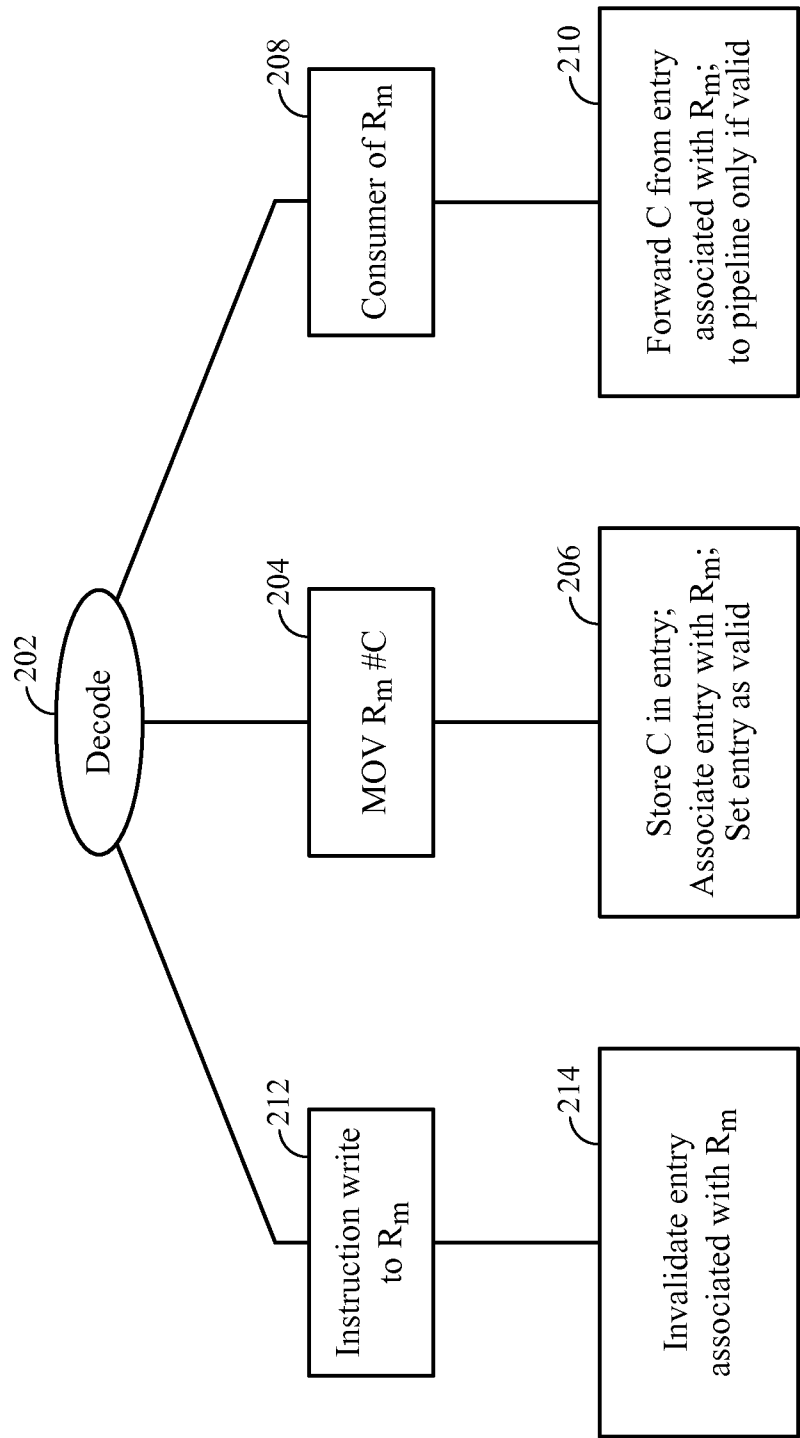
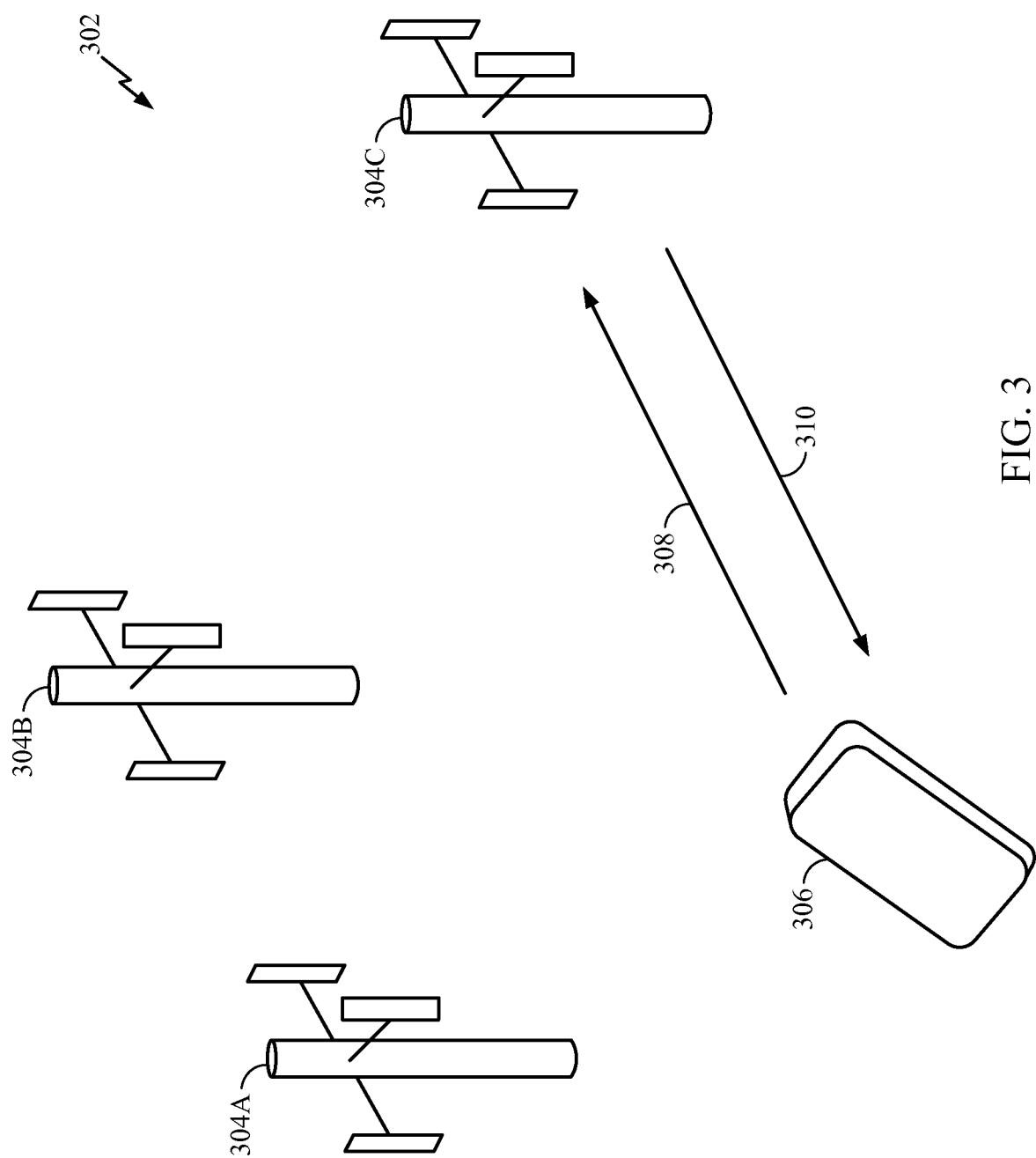


FIG. 2



# INTERNATIONAL SEARCH REPORT

International application No  
PCT/US2014/026907

## A. CLASSIFICATION OF SUBJECT MATTER

INV. G06F9/30 G06F9/38  
ADD.

According to International Patent Classification (IPC) or to both national classification and IPC

## B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

EPO-Internal, WPI Data

## C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 6 505 293 B1 (JOURDAN STEPHAN J [US] ET AL) 7 January 2003 (2003-01-07) column 6, line 40 - column 8, line 53; figure 2 -----	1-33
A	WO 2007/027671 A2 (SEARETE LLC [US]; GLEW ANDREW F [US]) 8 March 2007 (2007-03-08) paragraph [0033] paragraph [0044] - paragraph [0047] paragraph [0077] - paragraph [0078] paragraph [0099] -----	1-33
A	US 2010/106944 A1 (SYMES DOMINIC HUGO [GB] ET AL) 29 April 2010 (2010-04-29) paragraph [0337] - paragraph [0338]; figure 65 -----	1-33

☐ Further documents are listed in the continuation of Box C.

☒ See patent family annex.

### \* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search

28 May 2014

Date of mailing of the international search report

04/06/2014

Name and mailing address of the ISA/

European Patent Office, P.B. 5818 Patentlaan 2  
NL - 2280 HV Rijswijk  
Tel. (+31-70) 340-2040,  
Fax: (+31-70) 340-3016

Authorized officer

Moraiti, Marina

# INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No

PCT/US2014/026907

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 6505293	B1	07-01-2003	NONE
-----			
WO 2007027671	A2	08-03-2007	GB 2444455 A 04-06-2008
			KR 20080043378 A 16-05-2008
			WO 2007027671 A2 08-03-2007
-----			
US 2010106944	A1	29-04-2010	NONE
-----			