(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
17 November 2016 (17.11.2016)

WIPO | PCT

(10) International Publication Number
WO 2016/182588 A1

(51) International Patent Classification:
G06F 12/08 (2006.01) *G06F 13/16* (2006.01)

(21) International Application Number:
PCT/US2015/042792

(22) International Filing Date:
30 July 2015 (30.07.2015)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
14/710,837 13 May 2015 (13.05.2015) US

(71) Applicant: **APPLIED MICRO CIRCUITS CORPORATION** [US/US]; 215 Moffett Park Drive, Sunnyvale, California 94089 (US).

(72) Inventor: **SVENDSEN, Kjeld**; 905 Mangrove Ave., Sunnyvale, California 94086 (US).

(74) Agent: **TUROCY, Gregory**; Amin, Turocy & Watson, LLP, 57th Floor - Key Tower, 127 Public Square, Cleveland, Ohio 44114 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, 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).

Published:

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

(54) Title: PREFETCH TAG FOR EVICTION PROMOTION

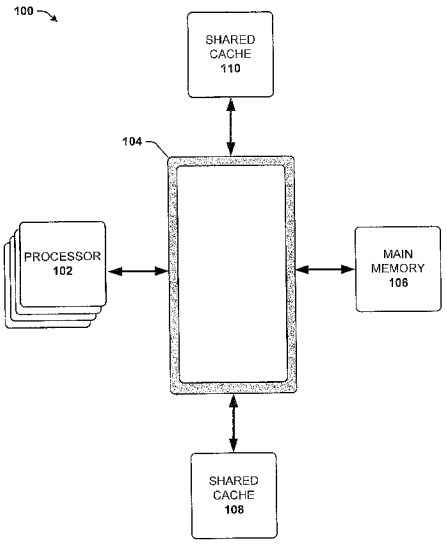

FIG. 1

(57) Abstract: System that prefetch data from a main memory to a cache and then evicts unused data to a lower level cache are described. The prefetching system prefetches data from a main memory to a cache, and data that is not immediately useable or is part of a data set which is too large to fit in the cache can be tagged for eviction to a lower level cache, which keeps the data available with a shorter latency than if the data had to be loaded from main memory again. This lowers the cost of prefetching useable data too far ahead and prevents cache trashing.

WO 2016/182588 A1

PREFETCH TAG FOR EVICTION PROMOTION

TECHNICAL FIELD

[0001] This disclosure relates to prefetching data to a cache from a main memory and evicting data to prevent cache trashing.

BACKGROUND

[0002] Modern microprocessors are much faster than the memory where the program is stored in, which means that the program instructions cannot be read fast enough to keep the microprocessor busy if the instructions are read from the main memory. Adding a cache, which is a small amount of very fast memory to each processor, can speed up processing time by providing faster access to needed instructions.

[0003] Prefetching is the process where anticipated instructions are loaded into the cache before being requested by the processor. Modern systems have evolved into multi-cache systems where each processor, or each core of the processor may have one or two levels of cache dedicated to each core/processor, and one or more additional cache levels that are shared among cores/processors. Each successive level of cache away from the core/processor may be larger but slower than the preceding cache levels. Prefetching from the main memory to a processor cache, such as level 1 or level 2 cache, can provide low latency access to the data, but since the size of the processor caches are small, data that is not used right away, or datasets which are larger than the size of the cache, can cause resource conflicts and upsets (cache trashing).

SUMMARY

[0004] In an embodiment, a cache prefetch system comprises an interconnect configured for communicably coupling a processor, a shared cache, and a main memory. The cache prefetch system can include a processor cache prefetcher configured for prefetching a set of data from the main memory via the interconnect to a processor cache, wherein the processor cache is associated with the processor, and wherein the processor cache tags a first portion of data from the set of data as unused and a second portion of data

from the set of data as used. The cache prefetch system can include a processor cache evictor configured for evicting the first portion of data to the shared cache via the interconnect and evicting the second portion of data to the main memory via the interconnect.

[0005] In another embodiment, a cache prefetch system can include a processor; and a memory that stores executable instructions that, when executed by the processor, facilitate performance of operations. The operations can include prefetching a set of data from a system memory to a processor cache, associated with the processor and tagging a first portion of data of the set of data with an indication that the first portion of data is executed data. The operations can include tagging a second portion of data of the set of data with an indication that the second portion of data is likely to be executed data and evicting the second portion of data to a shared cache.

[0006] In another embodiment, a method comprises prefetching a set of data from a system memory to a processor cache, associated with the processor. The method can include tagging a first portion of data of the set of data with an indication that the first portion of data is used data. The method can include tagging a second portion of data of the set of data with an indication that the second portion of data is unused data. The method can include evicting the second portion of data to a shared cache.

BRIEF DESCRIPTION OF THE DRAWINGS

[0007] **FIG. 1** is a block diagram illustrating an embodiment of a system for prefetching with a ring architecture interconnect.

[0008] **FIG. 2** is a block diagram illustrating an embodiment of a prefetching system.

[0009] **FIG. 3** is a block diagram illustrating an embodiment of a prefetching system.

[0010] **FIG. 4** is a block diagram illustrating an embodiment of a prefetching system.

[0011] **FIG. 5** is a block diagram illustrating an embodiment of a prefetching system.

[0012] **FIG. 6** is a block diagram illustrating an embodiment of a processor with a prefetching cache system that evicts data to prevent cache trashing.

[0013] **FIG. 7** is a block diagram illustrating an embodiment of a processor with a prefetching cache system that evicts data to prevent cache trashing.

[0014] **FIG. 8** illustrates a flow diagram of an embodiment of a method for prefetching and evicting unused data to prevent cache trashing.

[0015] **FIG. 9** illustrates a block diagram of an example electronic computing environment.

[0016] **FIG. 10** illustrates a block diagram of an example data communication network.

DETAILED DESCRIPTION

[0017] Various embodiments provide for a system that prefetches data from a main memory to a cache and then evicts unused data to a lower level cache. The prefetching system prefetches data from a main memory to a cache, and data that is not immediately useable or is part of a data set which is too large to fit in the cache can be tagged for eviction to a lower level cache, which keeps the data available with a shorter latency than if the data had to be loaded from main memory again. This lowers the cost of prefetching useable data too far ahead and prevents cache trashing.

[0018] Data can be prefetched well before it may be used, as loads from slower memories (DDR, disks) tend to be spatial and temporal in nature. Prefetching takes advantage of the spatial nature of loads since related data may be located physically or logically near the data that is being requested by the processor. Thus, for address streams, there is a high likelihood that a load to a memory address A will be followed soon by a load to memory address A+N, where N can be any arbitrary integer, and thus loads likely occur in strides to addresses A+N*m where m is another arbitrary signed integer.

[0019] By prefetching related data as well as currently requested data, future loading of data from the main memory or from storage can be avoided. Since processor level caches, such as level 2 caches are not very large, data should not be stored there unless it is next in line for execution or will be

processed very soon. Therefore, data that is not immediately next in line can be tagged as unused data, and can be evicted to a lower level cache, such as level 3 cache. Level 3 cache may be slower than level 2 cache, but it can be much larger, and can more feasibly retain data that may be used at a later time. When the data that was evicted to the level 3 cache is then required by the processor, the level 2 cache can prefetch that data from the level 3 cache must faster than prefetching the data from the main memory.

[0020] **FIG. 1**, a block diagram illustrating embodiment of a system 100 for prefetching with a ring architecture interconnect. Processor 102 can include one or more cores (shown in **FIG. 1** with 4 cores). Processor 102 can also include a processor level cache, or a cache associated with each core of processor 102. This cache can be a level 2 cache in some embodiments. The processor 102 can also be communicably coupled to one or more shared caches 108 and 110, and a main memory 106 via a ring interconnect 104. In systems with ring interconnects such as ring interconnect 104, the shared caches 108 and 110 can be shared among one or more processors (e.g., processor 102, etc) without being directly tied to each processor. The shared caches, level 3 caches in some embodiments, can thus be distributed among multiple cores and/or processors.

[0021] In an embodiment, processor 102 can include a processor level cache, or level 2 cache, for each core, or pair of cores in processor 102. Data that is executed by the processor 102 can be fetched from the level 2 cache to a level 1 cache, or even directly to a register associated with the processor 102. In order to ensure that the processor level cache on processor 102 has the data that the processor 102 will need for execution, and to avoid a cache miss, where the processor does not find the memory location in the cache, the processor cache can prefetch data from the main memory 106 via the ring interconnect 104. Using predictive heuristics, the prefetcher in the processor cache can predict which data is likely to be used next using the A+N*m algorithm described above, and preload the data into the processor cache so that the data is available when the processor 102 requires the data.

[0022] Since related data may be likely to be either physically or logically close to the requested data, or related in address space, the related data can be prefetched in addition to the requested data, since the related data may be

requested by the processor at a later time. If the dataset is too large for the level 2 cache on the processor, or may not be used within a predetermined time period, the level 2 cache can mark that data as "prefetch only" or as "unused" which will promote the data for eviction to the shared cache 108 or 110. Data that may be completely unrelated can be evicted back the main memory 106, but if the data may possibly be used in the near future, or within a predetermined time period, the data can be tagged for eviction to the shared cache 108 or 110. The data can be tagged by setting an indicator bit on or off based on whether the data is to be evicted to the shared cache 108 or 110. In an embodiment, the data can be tagged based on an indication of a probability of future use by the processor. Based on the probability of future use, the level 3 cache, or the shared cache 108 or 110 can hold the data for a predetermined length of time before evicting to the main memory 106. Similarly, if the probability of future use within a predetermined period of time is very high, the processor cache may not tag the data for eviction.

[0023] When that data is requested by the processor at the later time, the latency to retrieve the data from the shared cache 108 or 110 will be much shorter than the latency retrieving the data from main memory 106.

[0024] It is to be appreciated that although reference has been made to level 2 and level 3 cache, the concepts herein that describe a prefetch system that tags prefetched data for eviction to a hierarchically lower cache system can be applied in other embodiments. For instance, data can be prefetched to a level 1 cache, and then evicted to a level 2 cache, and in other embodiments, level 3 and 4 caches can be used, or various combinations thereof.

[0025] **FIG. 2** illustrates a block diagram illustrating an example embodiment of a prefetching system 200. Processor 202 can include one or more cores (shown in **FIG. 2** with 4 cores). Processor 202 can also include a processor level cache 204 and in some embodiments, each core can include a cache like cache 204. The cache 204 can be a level 2 cache in some embodiments. The processor 202 can also be communicably coupled to an interconnect 210 and via the interconnect 210 to a shared cache 208 and a main memory 206. In an embodiment, the shared cache 208 can be shared among one or more processors (e.g., processor 202, etc) and/or cores without being directly tied to each processor. The shared cache 208, level 3 cache in

some embodiments, can thus be distributed among multiple cores and/or processors.

[0026] Data that is executed by the processor 202 can be fetched from the level 2 cache to a level 1 cache, or even directly to a register associated with the processor 202. In order to ensure that the processor level cache 204 on processor 202 has the data that the processor 202 will need for execution, and to avoid a cache miss, where the processor does not find the memory location in the cache, the processor cache can prefetch data from the main memory 206 via the interconnect 210. Using predictive heuristics, the prefetcher in the processor cache can predict which data is likely to be used next using the A+N*m algorithm described above, and send a request to main memory 206 via interconnect 210 to preload the data into the processor cache 204 so that the data is available when the processor 202 requires the data.

[0027] **FIG. 3** illustrates a block diagram of an embodiment of a prefetching system 300. After the processor 302 and prefetcher in the cache 304 requests data to be preloaded from main memory 306, the data can be sent from the main memory 306 to the processor 302 and cache 304 via the interconnect 310. In some embodiments, the data can be directly prefetched to the level 2 cache 304, skipping the shared cache 308 (level 3 cache), and in other embodiments, the data can first be prefetched to shared cache 308 and then prefetched from shared cache 308 to cache 304. The data that is prefetched by the cache 304 can be data that is being requested by the processor 302 as well as other data that may be related to the data being requested. The related data can be data that is related in address space, or located physically or logically near the requested data, and by prefetching all the possibly related data can be more efficient than prefetching, in multiple operations, the requested data.

[0028] **FIG. 4** illustrates another block diagram of an embodiment of a prefetching system 400. After the processor 402 and cache 404 have prefetched a set of related data from a main memory 406, the cache 404 can evict a portion of the data to a shared cache 408 via an interconnect 410. Since related data may be likely to be either physically or logically close to the requested data, or related in address space, the related data can be prefetched in addition to the requested data since the related data may be requested by the

processor at a later time. If the dataset is too large for the cache 404 on the processor 402, or may not be used within a predetermined time period, the cache 404 can mark that data as "prefetch only" or as "unused" which will promote the data for eviction to the shared cache 408. Data that may be completely unrelated can be evicted back the main memory 406, but if the data may possibly be used in the near future, or within a predetermined time period, the data can be tagged for eviction to the shared cache 408. The data can be tagged by setting an indicator bit on or off based on whether the data is to be evicted to the shared cache 408. In an embodiment, the data can be tagged based on an indication of a probability of future use by the processor 402. Based on the probability of future use, the level 3 cache, or the shared cache 408 can hold the data for a predetermined length of time before evicting to the main memory 406. Similarly, if the probability of future use within a predetermined period of time is very high or above a predetermined probability, the processor cache 404 may not tag the data for eviction.

[0029] **FIG. 5** illustrates a block diagram of an embodiment of a prefetching system 500. After the tagged data has been evicted to the shared cache 508, if the processor 502 requests the evicted data, or a prefetcher on cache 504 determines that the processor 502 will soon process/execute the related data, the data can be retrieved from the shared cache 508 via the interconnect 510. In an embodiment, if the data has already been evicted back to the main memory 506, the cache 504 can load the data from the main memory 506 via the interconnect 510.

[0030] **FIG. 6** illustrates a block diagram illustrating an embodiment of a processor with a prefetching cache system 600 that evicts data to prevent cache trashing. Processor 602 can include one or more cores and a cache 604. In some embodiments, each core, or pair of cores may have a corresponding cache. In some embodiments, cache 604 can be a level 2 cache. The processor 602 can also be communicably coupled to a shared cache 612, and a main memory 614 (DDR, disk, etc) via an interconnect 610. In systems with ring interconnects such as interconnect 610, the shared cache 612 can be shared among one or more processors (e.g., processor 602, etc) without being directly tied to each processor. The shared cache 612, level 3 caches in some embodiments, can thus be distributed among multiple cores and/or processors.

[0031] In an embodiment, the cache 604 can include a prefetch component 606 (prefetcher) and an eviction component 608 (evictor). The prefetch component can be configured to prefetch a set of data from the main memory 614 via the interconnect 610 to the cache 604, wherein the processor cache 604 is associated with the processor 602. The prefetch component 606 can tags a first portion of data from the set of data as unused and a second portion of data from the set of data as used.

[0032] The eviction component 608 can be configured to evict the first portion of data to the shared cache 612 via the interconnect 610 and evict the second portion of data to the main memory 614 via the interconnect 610.

[0033] **FIG. 7** illustrates a block diagram of an embodiment of a processor with a prefetching cache system 700 that evicts data to prevent cache trashing. The processor 702 can include a cache 704 that has a tagging component 706 and a prediction component 708. The tagging component 706 can tag data that is prefetched by the cache 704 to indicate that the data is unused, or is likely to be used in the future.

[0034] Since data related to the processor requested data may be likely to be either physically or logically close to the requested data, or related in address space, the related data can be prefetched in addition to the requested data, since the related data may be requested by the processor 702 at a later time. If the dataset is too large for the cache 704 on the processor, or may not be used within a predetermined time period, the cache 704 can tag that data as "prefetch only" or as "unused" which will promote the data for eviction to a level 3 cache or a lower level cache. Data that may be completely unrelated can be evicted back a main memory, but if the data may possibly be used in the near future, or within a predetermined time period, the data can be tagged for eviction to the shared cache. The data can be tagged by the tagging component 706 by setting an indicator bit on or off based on whether the data is to be evicted to the shared cache. In an embodiment, the data can be tagged based on an indication of a probability of future use by the processor. The prediction component 708 can determine a likelihood that the data is to be used by the processor 702 within a predetermined time period, and based on the probability of future use, the level 3 cache, or the shared cache can hold the data for a predetermined length of time before evicting to the main memory.

The tagging component 706 can also set an indicator on the data indicating the relative or absolute probability of future use.

[0035] In view of the example systems described above, methods that may be implemented in accordance with the described subject matter may be better appreciated with reference to the flow chart of **FIG. 8**. While the method is shown and described as a series of blocks, the claimed subject matter is not limited by the order of the blocks, as some blocks may occur in different orders and/or concurrently with other blocks from what is depicted herein. Not all illustrated blocks may be required to implement the methods described hereinafter.

[0036] **FIG. 8** is a flow diagram of an embodiment of a method for prefetching and evicting unused data to prevent cache trashing. Methodology 700 can start at 702, where the method include prefetching a set of data from a system memory to a processor cache, associated with the processor. The prefetching can be performed by the prefetcher in the processor cache which can be a level 2 cache in some embodiments. The prefetching loads not just the data at a memory address indicated by the processor, but also data at memory addresses related to the requested data. Using predictive heuristics, the prefetcher can predict which data is likely to be used in the future, which can include data in a similar address space, or data physically or logically near to the requested data. This related data can be preloaded into the cache to allow faster access by the processor to the data.

[0037] At 704, the method can include tagging a first portion of data of the set of data with an indication that the first portion of data is used data. If data is used by the processor and if it is not likely to be used again in the near future, the data can be tagged for eviction back to the main memory or to disk. Similarly, at 706, the method can include tagging a second portion of data of the set of data with an indication that the second portion of data is likely to be executed data and at 708 where the method includes evicting the second portion of data to a shared cache. Data that may not be used right away, but may be used in the near future can be sent to a shared cache, (e.g., level 3 cache). This shared cache provides quicker access to the data than from main memory, and evicting it, allows the cache at the processor (level 2 cache) to

retain data that might be used more immediately or sooner than the evicted data.

[0038] The techniques described herein can be applied to any device where it is desirable to facilitate the execution of prefetching and evicting to avoid cache trashing. Handheld, portable and other computing devices and computing objects of all kinds are contemplated for use in connection with the various embodiments, i.e., anywhere that a device may wish to share computing resources with a plurality of guest devices or virtual machines. Accordingly, the below general purpose remote computer described below in **FIG. 9** is one example, and the disclosed subject matter can be implemented with any client having network/bus interoperability and interaction. Thus, the disclosed subject matter can be implemented in an environment of networked hosted services in which very little or minimal client resources are implicated, *e.g.*, a networked environment in which the client device serves merely as an interface to the network/bus, such as an object placed in an appliance.

[0039] **FIG. 9** illustrates an example of a computing system environment 900 in which some aspects of the disclosed subject matter can be implemented, although the computing system environment 900 is one example of a computing environment for a device.

[0040] **FIG. 9** is an exemplary device for implementing the disclosed subject matter includes a general-purpose computing device in the form of a computer 910. Components of computer 910 may include a processing unit 920, a system memory 930, and a system bus 921 that couples various system components including the system memory to the processing unit 920.

[0041] Computer 910 includes a variety of computer readable media. Computer readable media can be any available media that can be accessed by computer 910. The system memory 930 may include computer storage media in the form of volatile and/or nonvolatile memory such as read only memory (ROM) and/or random access memory (RAM).

[0042] The computer 910 can operate in a networked or distributed environment using logical connections to one or more other remote computer(s), such as remote computer 970, which can in turn have media capabilities different from device 910. The logical connections depicted in **FIG. 9** include a network 971, such local area network (LAN) or a wide area network

(WAN), but can also include other networks/buses, either wired or wireless. When used in a LAN networking environment, the computer 910 can be connected to the LAN 971 through a network interface or adapter. When used in a WAN networking environment, the computer 910 can typically include a communications component, such as a modem, or other means for establishing communications over the WAN, such as the Internet.

[0043] **FIG. 10** is a schematic diagram of an exemplary networked or distributed computing environment. The distributed computing environment comprises computing objects 1010, 1012, etc. and computing objects or devices 1020, 1022, 1024, 1026, 1028, etc., which may include programs, methods, data stores, programmable logic, etc., as represented by applications 1030, 1032, 1034, 1036, 1038 and data store(s) 1040.

[0044] Each computing object 1010, 1012, etc. and computing objects or devices 1020, 1022, 1024, 1026, 1028, etc. can communicate with one or more other computing objects 1010, 1012, etc. and computing objects or devices 1020, 1022, 1024, 1026, 1028, etc. by way of the communications network 1042. Communications network 1042 may comprise other computing objects and computing devices that provide services to the system of **FIG. 10**, and/or may represent multiple interconnected networks. A host of network topologies and network infrastructures, such as client/server, peer-to-peer, or hybrid architectures, can be utilized. In a client/server architecture, particularly a networked system, a client is usually a computer that accesses shared network resources provided by another computer, e.g., a server.

[0045] Both a process executed from memory and the processor can be a component. As another example, an architecture can include an arrangement of electronic hardware (*e.g.*, parallel or serial transistors), processing instructions and a processor, which implement the processing instructions in a manner suitable to the arrangement of electronic hardware.

[0046] The disclosed subject matter can be implemented as a method, apparatus, or article of manufacture using typical manufacturing, programming or engineering techniques to produce hardware, firmware, software, or any suitable combination thereof to control an electronic device to implement the disclosed subject matter. Computer-readable media can include hardware media, software media, non-transitory media, or transport media.

CLAIMS

What is claimed is:

1. A cache prefetch system, comprising:

an interconnect configured for communicably coupling a processor, a shared cache, and a main memory;

a processor cache prefetcher configured for prefetching a set of data from the main memory via the interconnect to a processor cache, wherein the processor cache is associated with the processor, and wherein the processor cache prefetcher tags a first portion of data from the set of data as unused and a second portion of data from the set of data as used; and

a processor cache evictor configured for evicting the first portion of data to the shared cache via the interconnect and evicting the second portion of data to the main memory via the interconnect.

2. The cache prefetch system of claim 1, wherein

the interconnect comprises a ring interconnect;

the processor cache comprises a level 2 cache, and the shared cache comprises a level 3 cache; and

the set of data comprises a first data that is executed by the processor, and a second data that is associated with the first data.

3. The cache prefetch system of claim 1, wherein the processor cache prefetcher is further configured for tagging the first portion of data and the second portion of data with a indication of a probability of future use by the processor.

4. The cache prefetch system of claim 1, wherein the processor cache prefetcher is further configured for prefetching the first portion of data from the shared cache in response to the processor requesting the first portion of data.

5. A cache prefetch system comprising:

a processor; and

a memory that stores executable instructions that, when executed by the processor, facilitate performance of operations, comprising:

prefetching a set of data from a system memory to a processor cache, associated with the processor;

tagging a first portion of data of the set of data with an indication that the first portion of data is executed data;

tagging a second portion of data of the set of data with an indication that the second portion of data is likely to be executed data; and

evicting the second portion of data to a shared cache.

6. The cache prefetch system of claim 5, wherein the operations further comprise:

evicting the first portion of data to the system memory; and

prefetching the evicted second portion of data from the shared cache to the processor cache in response to receiving a request to execute the second portion of data from the processor.

7. The cache prefetch system of claim 5, wherein

the processor, the system memory, and the shared cache are communicably coupled via ring interconnect;

the second portion of data is likely to be executed data within a predetermined period of time; and

the first portion of data and the second portion of data are related to each other based on a logical and physical proximity to each other.

8. A caching method, comprising:

prefetching a set of data from a system memory to a processor cache, the processor cache associated with the processor;

tagging a first portion of data of the set of data with an indication that the first portion of data is used data;

tagging a second portion of data of the set of data with an indication that the second portion of data is unused data; and

evicting the second portion of data to a shared cache.

9. The caching method of claim 8, further comprising:

evicting the first portion of data to the system memory; and

prefetching the evicted second portion of data from the shared cache to the processor cache in response to receiving a request to execute the second portion of data from the processor.

10. The caching method of claim 8, wherein

the prefetching and the evicting are via a ring interconnect coupling the processor, the system memory, and the shared cache; and

the second portion of data is data that is likely to be used within a predetermined period of time.

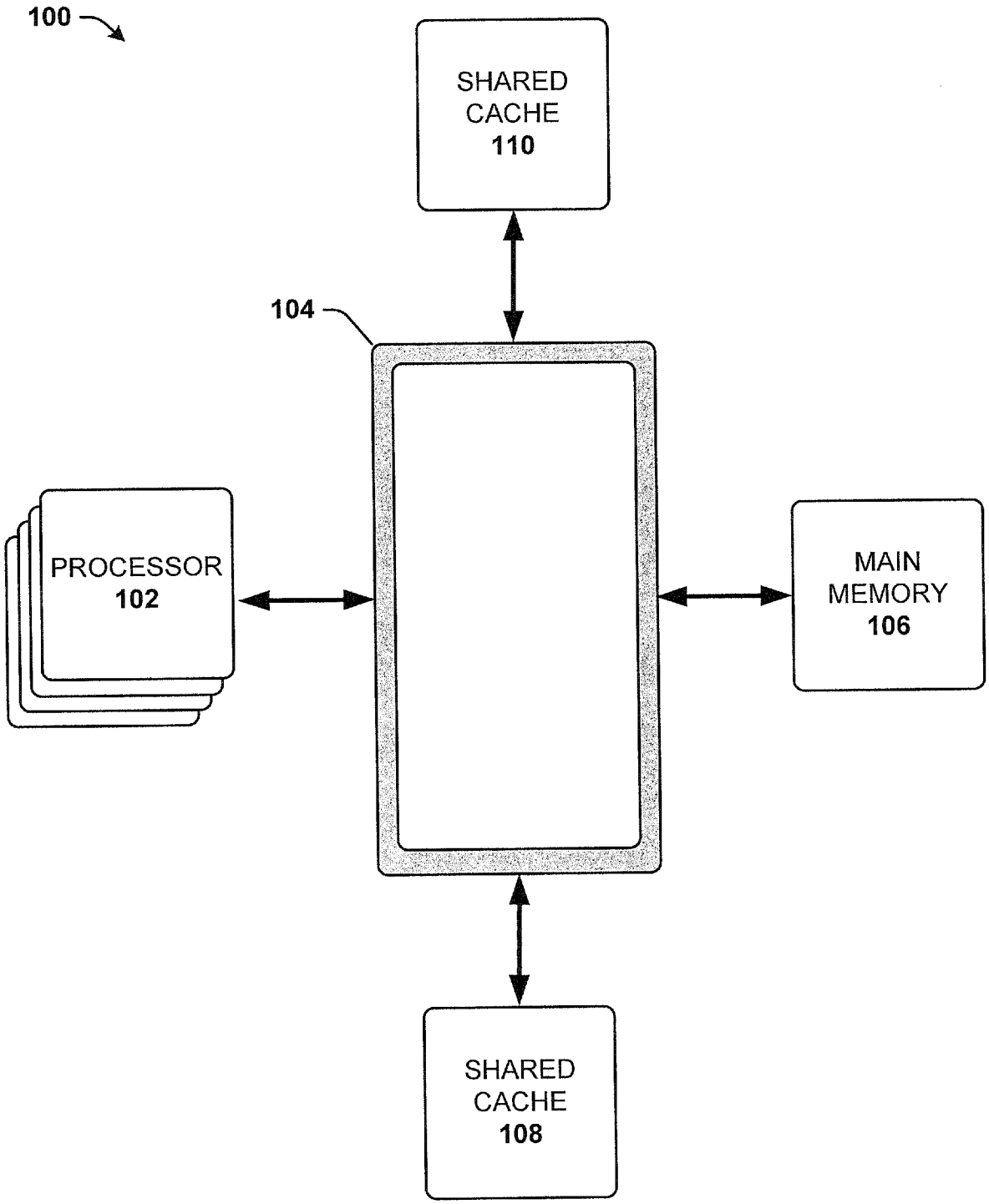
100
SHARED CACHE 110
104
PROCESSOR 102
MAIN MEMORY 106
SHARED CACHE 108

FIG. 1

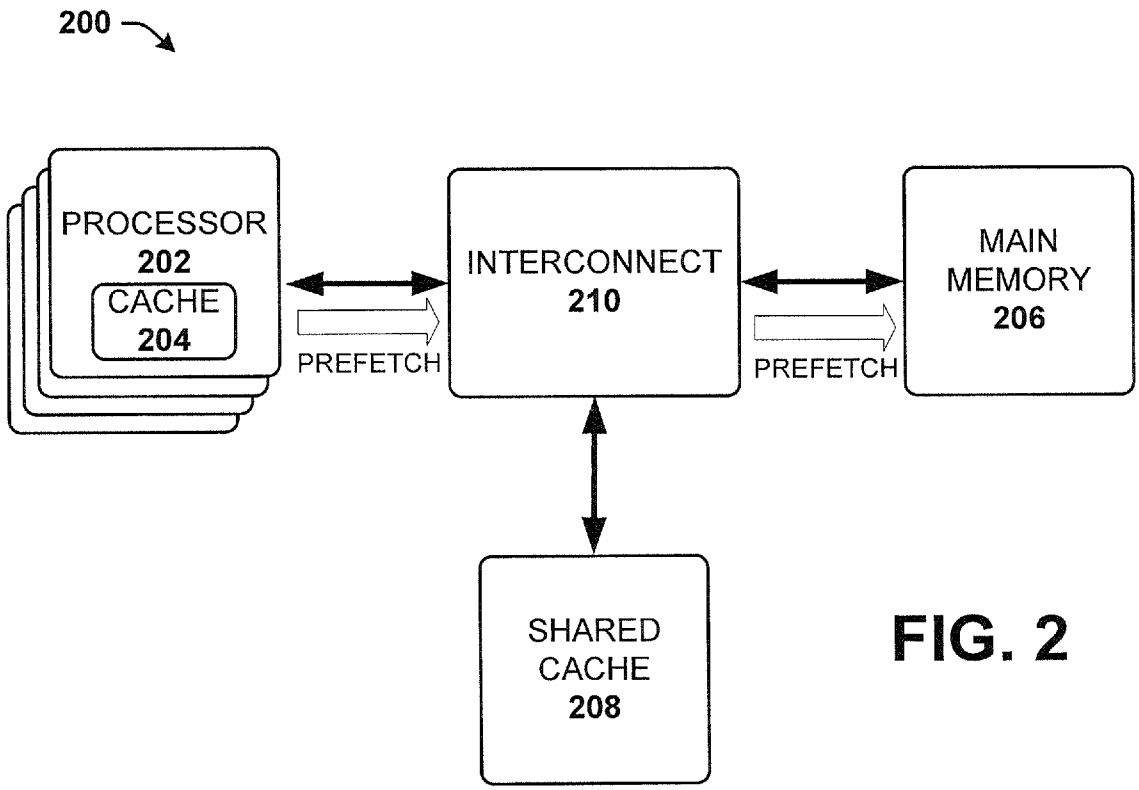
200
PROCESSOR
202
CACHE
204
PREFETCH
INTERCONNECT
210
PREFETCH
MAIN
MEMORY
206
SHARED
CACHE
208

FIG. 2

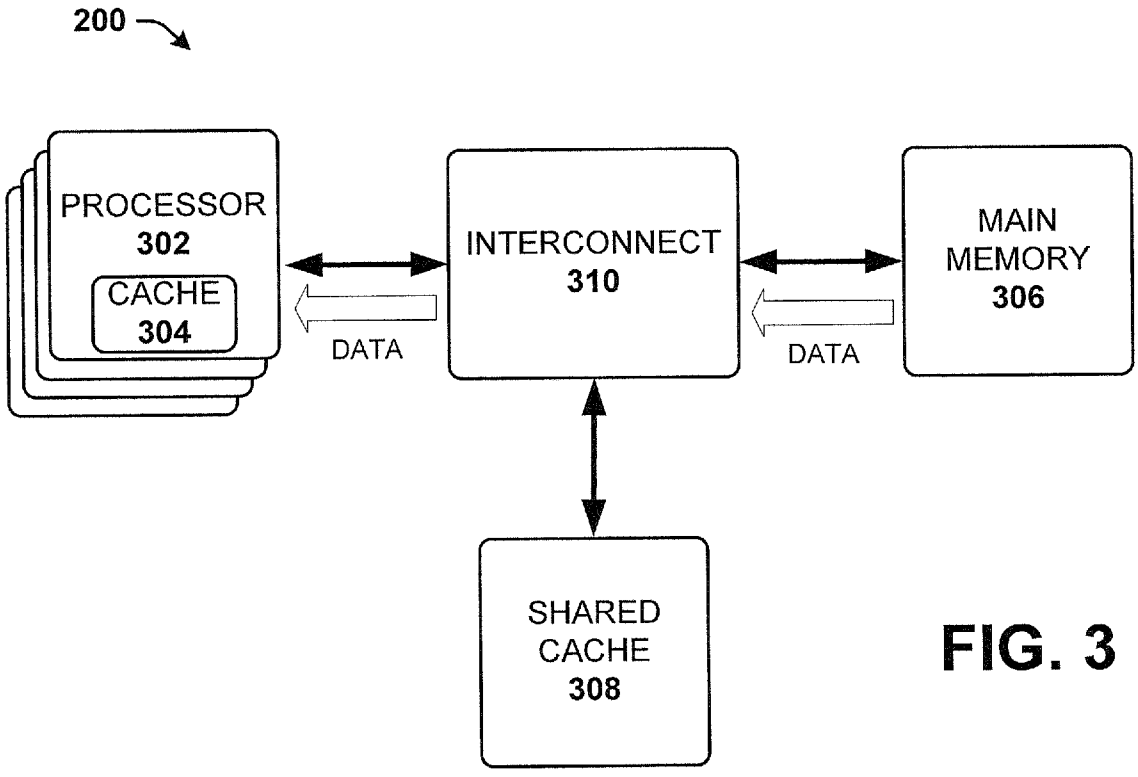
200
PROCESSOR
302
CACHE
304
DATA
INTERCONNECT
310
DATA
MAIN
MEMORY
306
SHARED
CACHE
308

FIG. 3

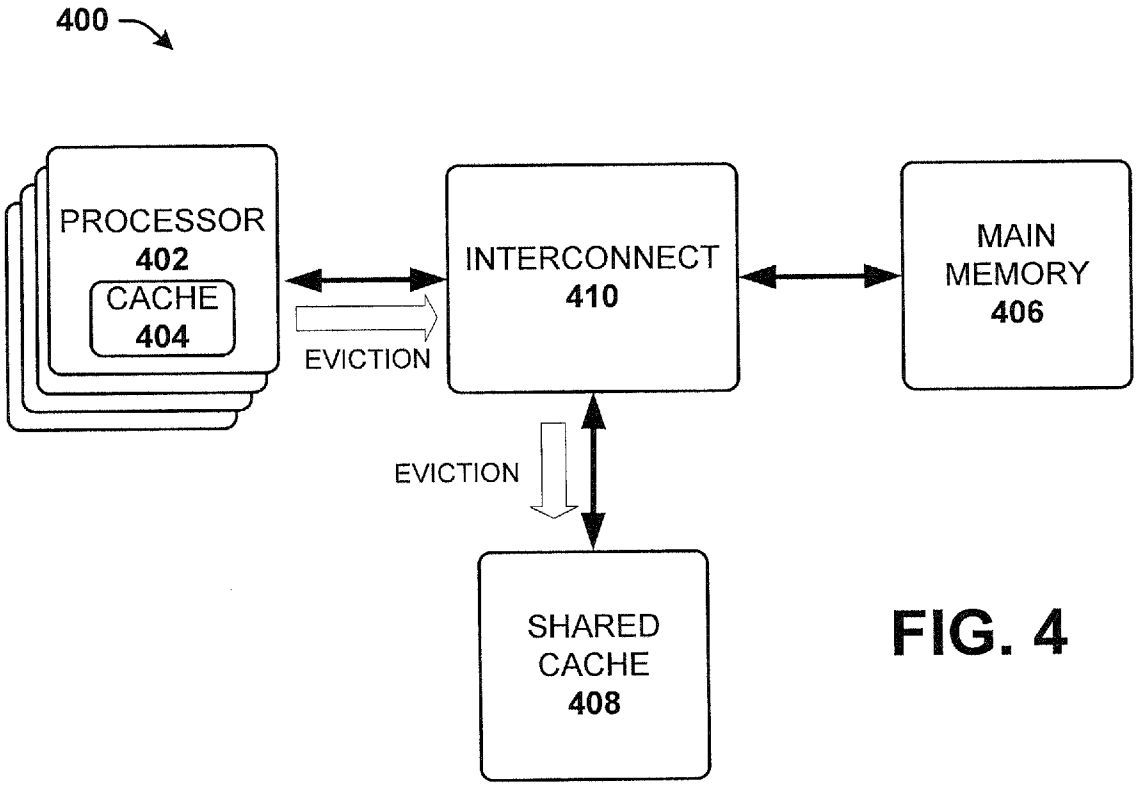
400
PROCESSOR
402
CACHE
404
EVICTION
INTERCONNECT
410
MAIN
MEMORY
406
EVICTION
SHARED
CACHE
408

FIG. 4

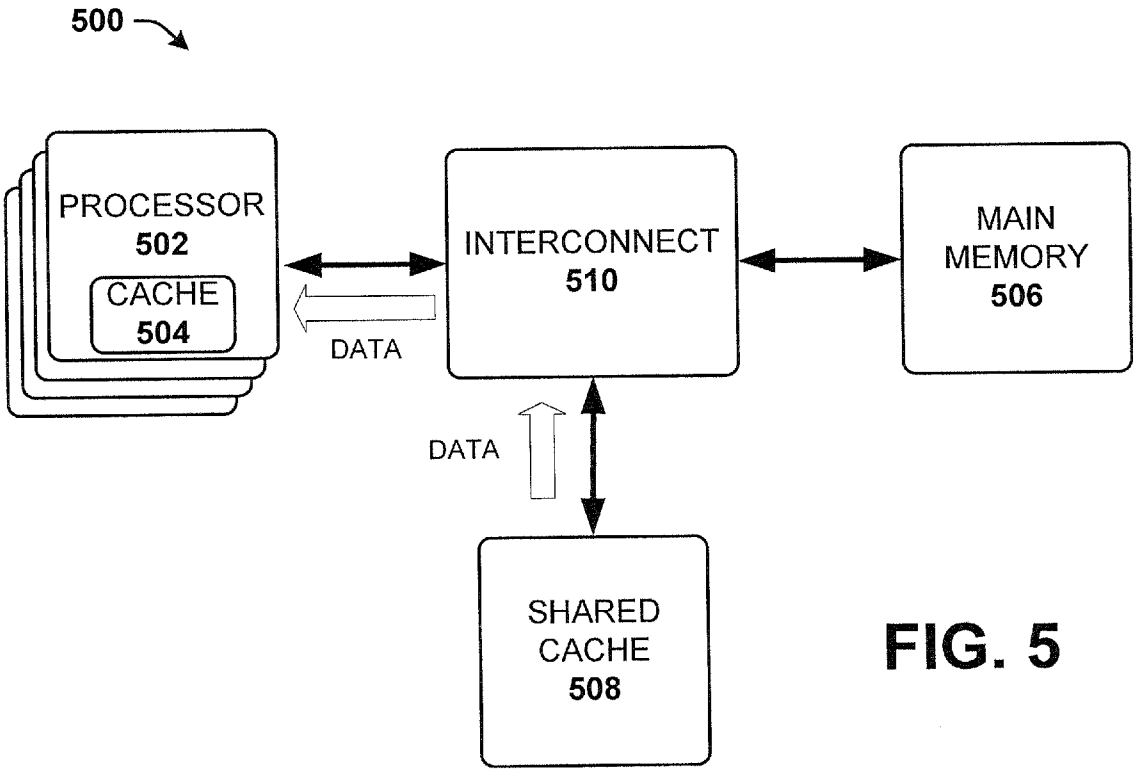
500
PROCESSOR
502
CACHE
504
DATA
INTERCONNECT
510
MAIN
MEMORY
506
DATA
SHARED
CACHE
508

FIG. 5

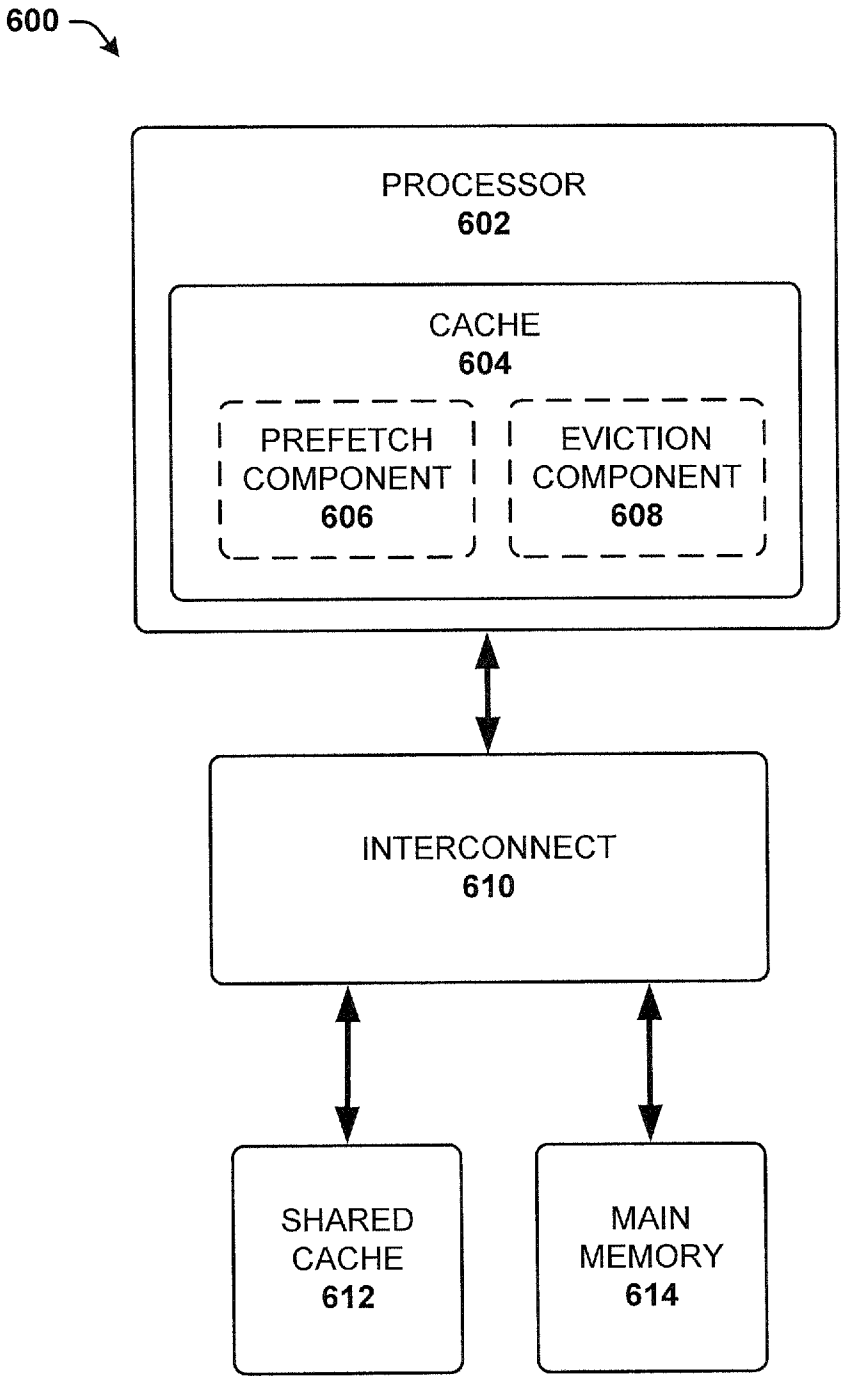
600
PROCESSOR
602
CACHE
604
PREFETCH COMPONENT
606
EVICTION COMPONENT
608
INTERCONNECT
610
SHARED CACHE
612
MAIN MEMORY
614

FIG. 6

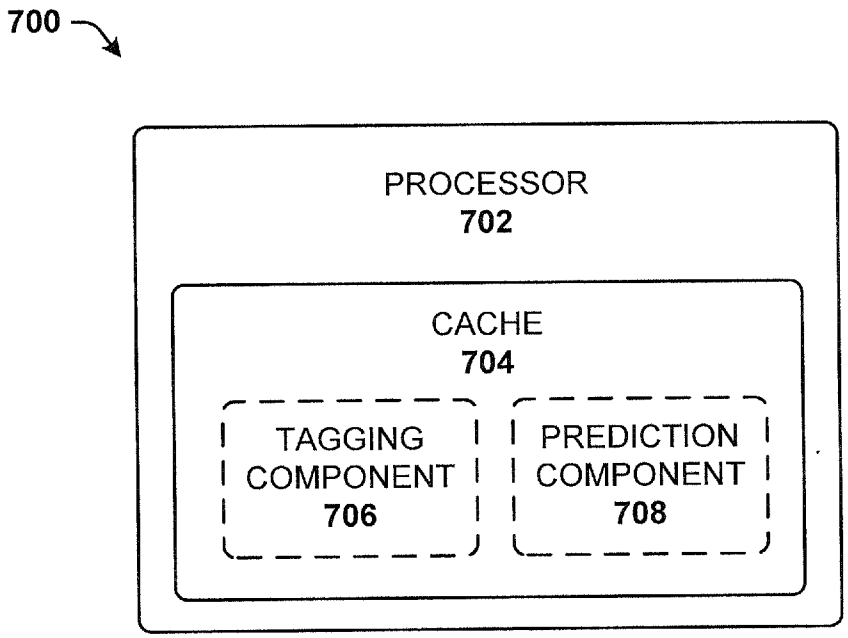
700
PROCESSOR
702
CACHE
704
TAGGING COMPONENT
706
PREDICTION COMPONENT
708

FIG. 7

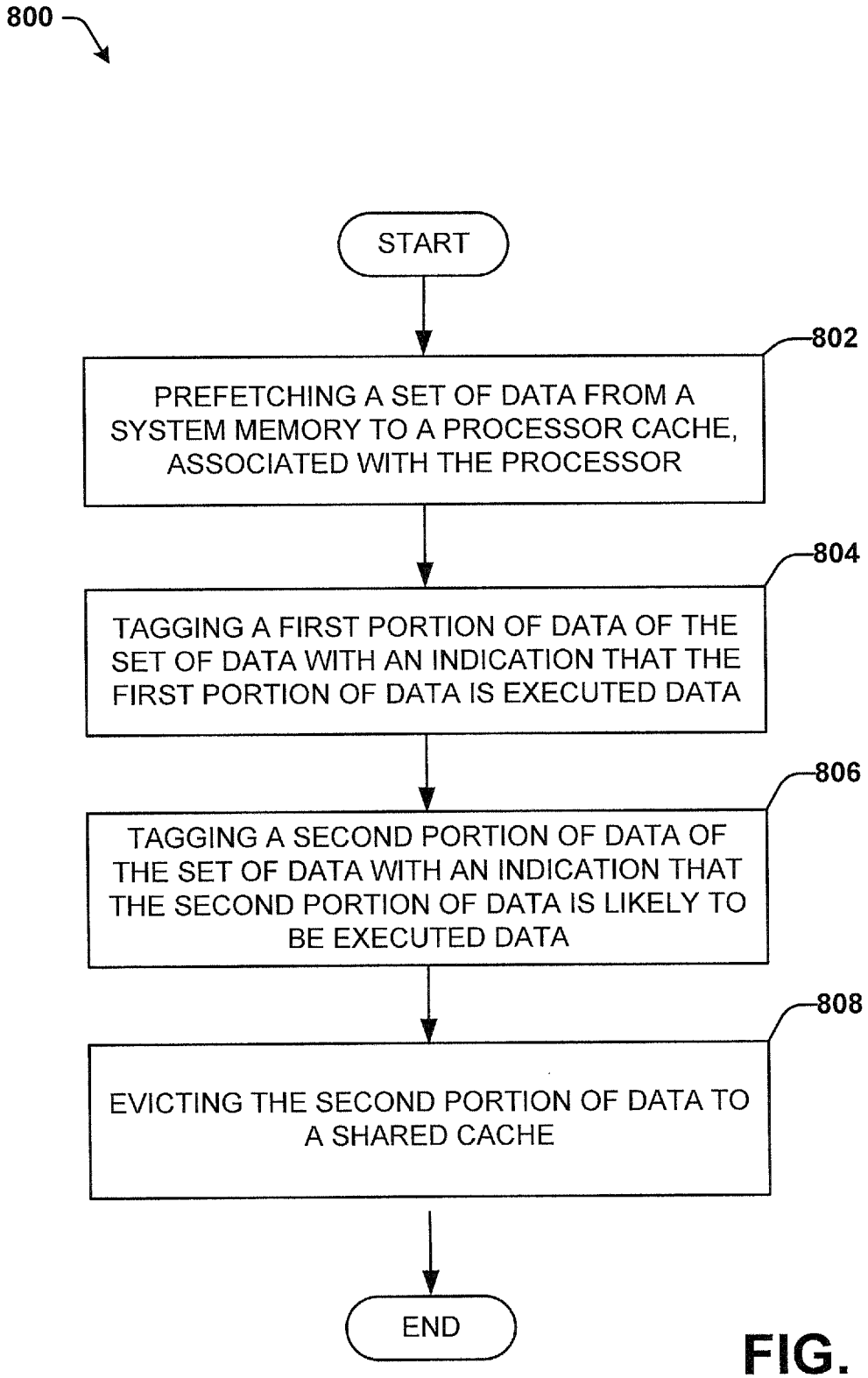
800
START
802
PREFETCHING A SET OF DATA FROM A SYSTEM MEMORY TO A PROCESSOR CACHE, ASSOCIATED WITH THE PROCESSOR
804
TAGGING A FIRST PORTION OF DATA OF THE SET OF DATA WITH AN INDICATION THAT THE FIRST PORTION OF DATA IS EXECUTED DATA
806
TAGGING A SECOND PORTION OF DATA OF THE SET OF DATA WITH AN INDICATION THAT THE SECOND PORTION OF DATA IS LIKELY TO BE EXECUTED DATA
808
EVICTING THE SECOND PORTION OF DATA TO A SHARED CACHE
END

FIG. 8

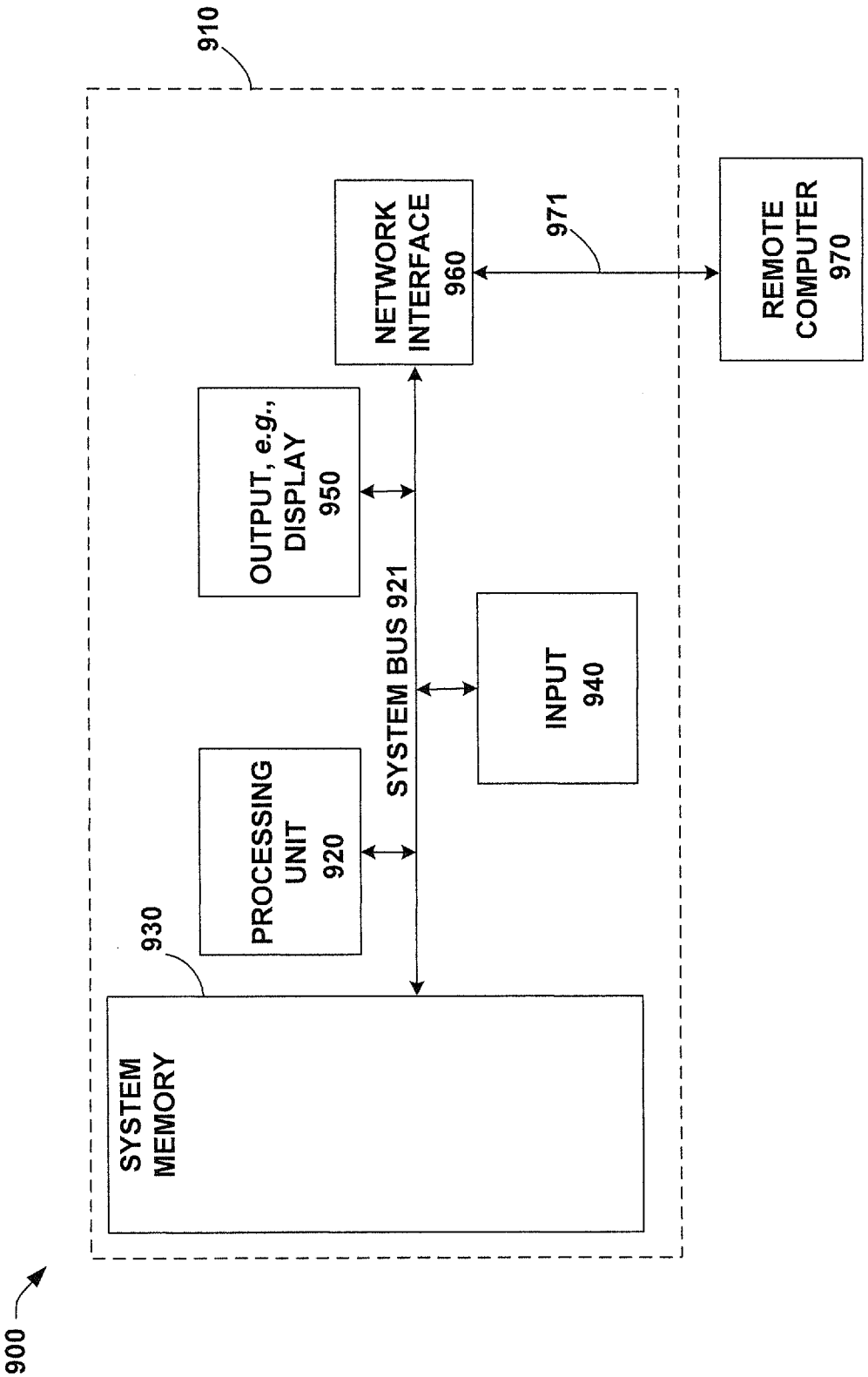
900
910
SYSTEM MEMORY
930
PROCESSING UNIT 920
SYSTEM BUS 921
OUTPUT, e.g., DISPLAY 950
INPUT 940
NETWORK INTERFACE 960
971
REMOTE COMPUTER 970

FIG. 9

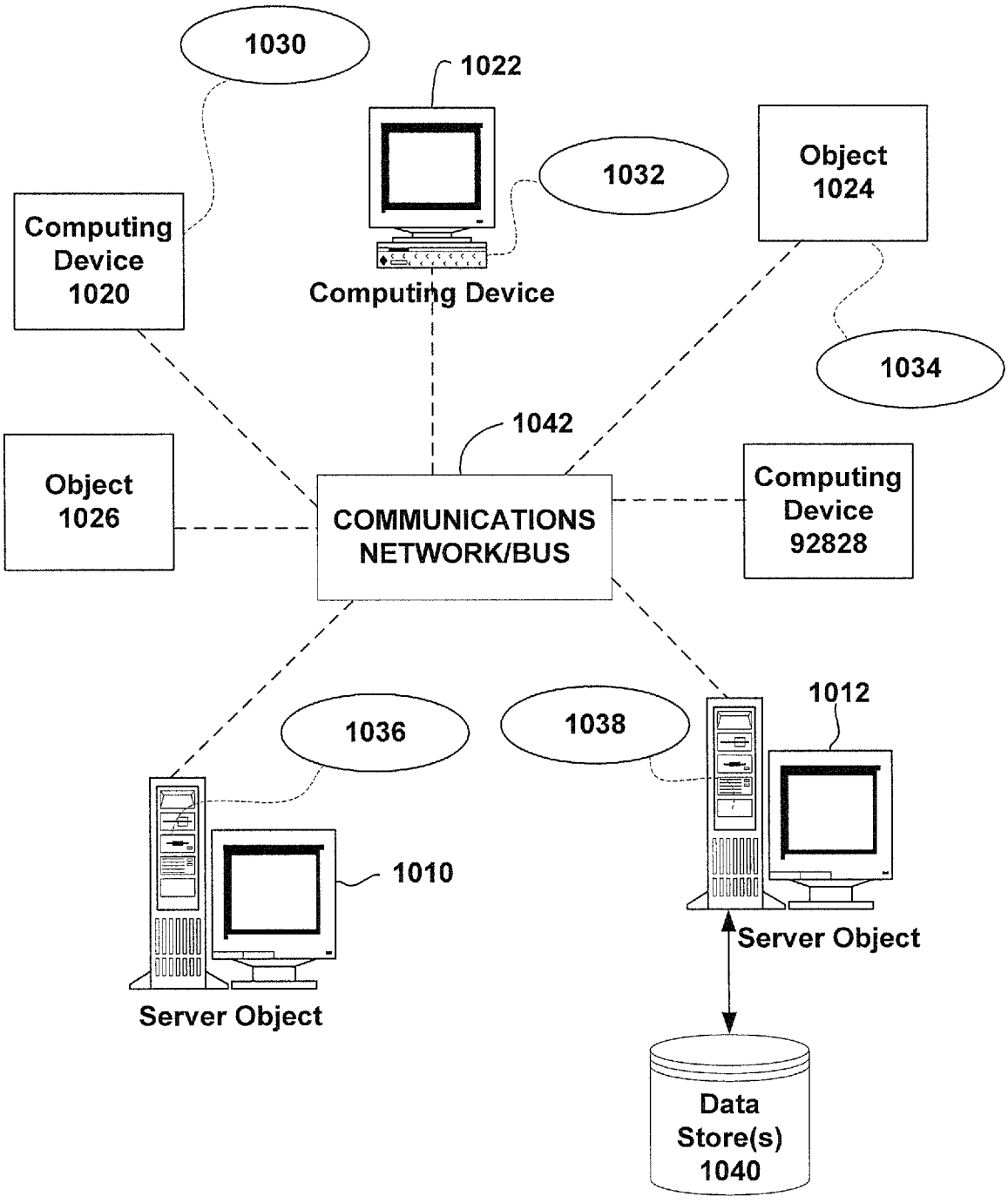
1030
1022
1032
Object 1024
Computing Device 1020
Computing Device
1034
1042
Object 1026
COMMUNICATIONS NETWORK/BUS
Computing Device 92828
1036
1038
1012
1010
Server Object
Server Object
Data Store(s) 1040

FIG. 10

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2015/042792

A. CLASSIFICATION OF SUBJECT MATTER

G06F 12/08(2006.01)i, G06F 13/16(2006.01)i

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 12/08; G06F 12/12; G06F 12/00; G06F 13/16

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
Korean utility models and applications for utility models
Japanese utility models and applications for utility models

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
eKOMPASS(KIPO internal) & Keywords: cache, prefetch, data set, tag, eviction, interconnect

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2014-0181402 A1 (ADVANCED MICRO DEVICES, INC.) 26 June 2014 See paragraphs [0002]-[0005], [0019]-[0023], [0030]-[0039], [0044], [0065]; claim 17; and figures 1, 4.	1,3-6,8-9
Y		2,7,10
Y	US 2011-0113199 A1 (PUQI P. TANG et al.) 12 May 2011 See paragraphs [0013], [0043]; claim 13; and figures 1, 4.	2,7,10
A	US 2011-0072218 A1 (SRILATHA MANNE et al.) 24 March 2011 See paragraphs [0007], [0030]; and figure 1.	1-10
A	US 2014-0173203 A1 (ANDREW T. FORSYTH) 19 June 2014 See paragraphs [0010], [0029]; and figure 4.	1-10
A	EP 1612683 A2 (INTEL CORPORATION) 04 January 2006 See paragraphs [0009], [0017]; and figure 2.	1-10

☐ 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	Date of mailing of the international search report
12 February 2016 (12.02.2016)	**12 February 2016 (12.02.2016)**

Name and mailing address of the ISA/KR
International Application Division
Korean Intellectual Property Office
189 Cheongsa-ro, Seo-gu, Daejeon Metropolitan City, 35208, Republic of Korea
Facsimile No. +82-42-472-7140

Authorized officer
BYUN, Sung Cheal
Telephone No. +82-42-481-8262

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.
PCT/US2015/042792

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2014-0181402 A1	26/06/2014	None	
US 2011-0113199 A1	12/05/2011	US 08443151 B2	14/05/2013
		US 08924651 B2	30/12/2014
		US 2014-136795 A1	15/05/2014
US 2011-0072218 A1	24/03/2011	None	
US 2014-0173203 A1	19/06/2014	None	
EP 1612683 A2	04/01/2006	CN 100511184 C	08/07/2009
		CN 1728112 A	01/02/2006
		EP 1612683 A3	30/04/2008
		KR 10-2006-0049710 A	19/05/2006
		TW 285810 A	21/08/2007
		TW I285810 B	21/08/2007
		US 07558920 B2	07/07/2009
		US 2006-0004963 A1	05/01/2006