

(19) World Intellectual Property Organization
International Bureau



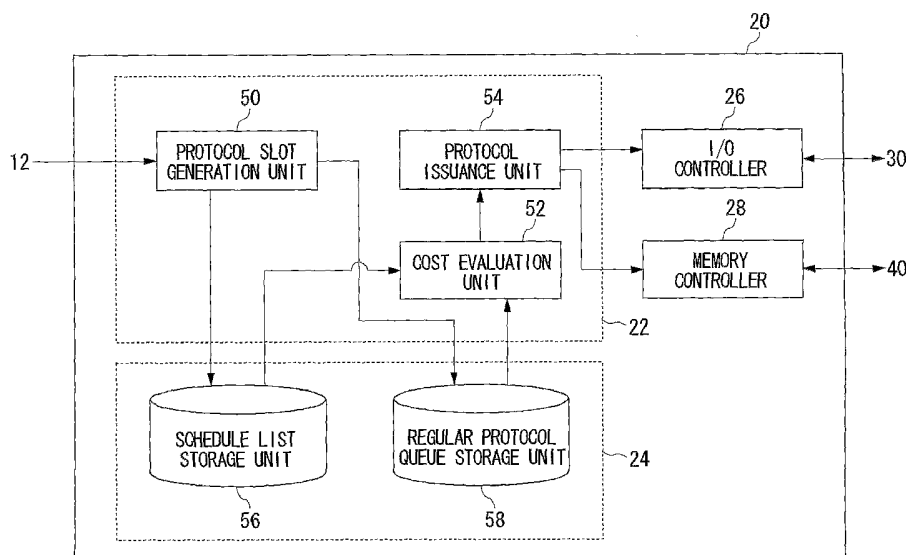
(43) International Publication Date
25 November 2004 (25.11.2004)

PCT

(10) International Publication Number
WO 2004/102392 A2

- (51) International Patent Classification⁷: **G06F 12/08**
- (21) International Application Number:
PCT/JP2004/005777
- (22) International Filing Date: 22 April 2004 (22.04.2004)
- (25) Filing Language: English
- (26) Publication Language: English
- (30) Priority Data:
2003-136387 14 May 2003 (14.05.2003) JP
- (71) Applicant: **SONY COMPUTER ENTERTAINMENT INC.** [JP/JP]; 2-6-21, Minami-Aoyama, Minato-ku, Tokyo 107-0062 (JP).
- (72) Inventor: **OHBA, Akio**; c/o SONY COMPUTER ENTERTAINMENT INC., 2-6-21, Minami-Aoyama, Minato-ku, Tokyo 107-0062 (JP).
- (74) Agent: **MORISHITA, Sakaki**; 2-11-12, Ebisu-Nishi, Shibuya-ku, Tokyo 150-0021 (JP).
- (81) Designated States (unless otherwise indicated, for every kind of national protection available): AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.
- (84) Designated States (unless otherwise indicated, for every kind of regional protection available): ARIPO (BW, GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IT, LU, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).
- Published:**
— without international search report and to be republished upon receipt of that report
- For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.*

(54) Title: PREFETCH COMMAND CONTROL METHOD, PREFETCH COMMAND CONTROL APPARATUS AND CACHE MEMORY CONTROL APPARATUS



(57) Abstract: A prefetch command control apparatus is provided. A protocol slot generation unit, when receiving from a CPU a prefetch command in which a utilization time of data to be prefetched is specified, generates a protocol slot based on the utilization time and registers it in a schedule list storage unit. When receiving a regular access command, the protocol slot generation unit registers a protocol slot in a regular protocol queue storage unit. A cost evaluation unit evaluates an issuance cost of the prefetch command based on a cache retention time of data to be prefetched and evaluates an issuance cost of the regular access command based on a resource occupation cost. A protocol issuance unit decides which command shall be issued, the prefetch command or regular access command, based on the evaluation result of these issuance costs.

WO 2004/102392 A2

DESCRIPTION

PREFETCH COMMAND CONTROL METHOD, PREFETCH COMMAND CONTROL
APPARATUS AND CACHE MEMORY CONTROL APPARATUS

5

TECHNICAL FIELD

The present invention relates to a cache control
technology, and it particularly relates to prefetch command
control method and apparatus and cache memory control
10 apparatus.

BACKGROUND TECHNOLOGY

One aspect of referencing main memory such as I/O and
DRAM by computer processor is locality. By taking advantage
15 of this locality and copying frequently referenced data
stored in main memory into the cache memory, which is a high
speed accessible small capacity internal memory, and
accessing the main memory through this cache memory, it is
possible to reduce latency for memory access. As the
20 processing performance of processors has improved, the
importance of cache memory is also becoming greater in terms
of high-speed data supply to the processor.

When the cache is hit, the processor can obtain
necessary data at the access speed of the cache memory. But,
25 in the case of a cache miss, the processor needs to stop
program execution while data is transferred from the main

memory to the cache memory. A cache miss occurs when a copy of the data in the main memory is not existent in the cache memory as a result of cached data being cached out due to a shortage of cache capacity or the data being accessed being
5 referenced for the first time. In the former case, the problem can be avoided by increasing the cache memory capacity. However, when the cache memory capacity increases, the cost increases so that it is difficult to significantly increase the capacity. In the latter case, the problem can
10 be avoided by using a method called prefetch, by which data expected to be used in the future are transferred to the cache memory in advance of execution.

Prefetch is an effective means to further reduce memory access time. However, a speculative prefetch may actually
15 lower the efficiency of the cache memory. This is because if prefetch is used too often when the cache memory capacity is small, formerly cached data is cached out by the prefetched data. Since there is no definite guideline for which situation and time a prefetch command shall be issued, while
20 many processors are prefetch command compatible, the actual state is that prefetch commands are often not used effectively.

DISCLOSURE OF THE INVENTION

25 The present invention has been made with the view to addressing the above-mentioned problems, and the object

thereof is to provide a prefetch command control method and apparatus to enable efficient memory access and a cache memory control apparatus.

According to one aspect of the present invention, a
5 prefetch command control method is provided. The method comprises generating a prefetch command for data with information specifying a utilization time of the data, and scheduling an issuance timing of the prefetch command according to the information specifying the utilization time.

10 The information specifying the utilization time in the prefetch command may include information specifying the data utilization start time or finish time by a relative time or by an absolute time. The issuance timing may be a time when a prefetch command is executed as an actual protocol for an
15 external resource. The scheduling of the issuance timing may be adjusting a time when the prefetch command shall be issued, adjusting an order in which a plurality of prefetch commands shall be issued one by one. The issuance timing may be adjusted between a prefetch command and a regular access
20 command such as a load command or a store command.

According to another aspect of the present invention, a prefetch command control apparatus is provided. The apparatus comprises a storage unit which stores a prefetch command for an external resource with information specifying
25 data utilization time added; a cost evaluation unit which evaluates a cost which accrues due to a prefetched data

staying in a cache by the specified utilization time when the stored prefetch command is issued; and an issuance unit which adjusts a timing when the stored prefetched command is issued for the external resource according to an evaluation result
5 by the cost evaluation unit.

The cost evaluation unit may compare the issuance cost of the prefetch command with an issuance cost of a regular access command stored in an access request queue for the external resource and the issuance unit may decide which of
10 the prefetch command and the regular access command shall be issued at a current time according to a comparison result by the cost evaluation unit. The cost evaluation unit may recalculate the issuance cost whenever the issuance unit issues a command and thereby update the issuance cost to be a
15 value at a current time.

When the issuance cost of the prefetch command is evaluated, a cost which accrues due to an issuance lag of a regular access command stored in an access request queue for the external resource or a risk that the data to be
20 prefetched is not prefetched by the data utilization time may be evaluated together.

According to still another aspect of the present invention, a cache memory control apparatus is provided. The apparatus comprises a cache memory which caches data obtained
25 from an external resource and provides the cached data to a processor; a scheduler which schedules an access protocol for

the external resource received from the processor and sets up the scheduled access protocol in a request queue for the external resource; and an external resource controller which reads the access protocol from the request queue and accesses
5 the external resource to provide data to the cache memory. The scheduler is the above-mentioned prefetch command control apparatus.

The scheduler may lock a cache of the prefetched data in the cache memory during a period of time for which the
10 prefetched data is utilized according to a utilization start time and a utilization finish time of the prefetched data included in the information specifying data utilization time, after the prefetch command is issued.

According to still another aspect of the present
15 invention, a computer program executable by a computer is provided. The program comprises analyzing a source code and converting the source code into an intermediate code; determining data to be prefetched based on the intermediate code, and then inserting an access command for the data as a
20 prefetch command into a position before an original execution sequence position, and adding information specifying a utilization time of the data in the prefetch command; and optimizing the intermediate code into which the prefetch command is inserted and generating an object code.

25 Moreover, any arbitrary replacement or substitution of the above-described structural components and the steps,

expressions replaced or substituted in part or whole between a method and an apparatus as well as addition thereof, and expressions changed to a system, a computer program, a storage medium, a transmission medium or the like are all effective as and are encompassed by the present invention.

This summary of the invention does not necessarily describe all necessary features, so that the invention may also be a sub-combination of these described features.

10 BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 is a block diagram of a processor system according to an embodiment of the present invention.

Fig. 2 is a flowchart describing a program compiling process according to an embodiment of the present invention.

15 Fig. 3(a) through Fig. 3(d) are diagrams describing examples of programs into which an extended prefetch command has been inserted according to the compiling process in Fig. 2.

Fig. 4 is a diagram describing how a cache of the cache memory of Fig. 1 is locked by issuance of an extended prefetch command.

Fig. 5 is a block diagram showing the bridge module of Fig. 1 in terms of its functions.

25 Fig. 6 is a flowchart describing a protocol slot generation process by the protocol slot generation unit of Fig. 5.

Fig. 7 is a diagram describing a schedule list of a just-in-time protocol stored in the schedule list storage unit of Fig. 5.

Fig. 8 is a diagram describing a regular protocol queue stored in the regular protocol queue storage unit of Fig. 5.

Fig. 9 is a flowchart showing an evaluation process for an issuance cost of access protocol by the cost evaluation unit of Fig. 5.

Fig. 10 is a diagram describing a schedule list, and a registration state of protocol slots in a regular protocol queue.

BEST MODE FOR CARRYING OUT THE INVENTION

The invention will now be described by reference to the preferred embodiments. This does not intend to limit the scope of the present invention, but to exemplify the invention.

Fig. 1 is a block diagram of a processor system according to the embodiment of the present invention. This processor system includes the processor module 10, the bridge module 20, the I/O module 30, and the memory module 40. The processor module 10 is connected to the internal bus 32; the I/O module 30 to the I/O bus 34; and the memory module 40 to the memory bus 36. The bridge module 20 bridges the internal bus 32 and the I/O bus 34, and controls data read/write on the I/O module 30 by the processor module 10. The bridge

module 20 bridges the internal bus 32 and the memory bus 36, and controls data read/write on the memory module 40 by the processor module 10.

The processor module 10 includes the CPU 12, the cache
5 memory 14, and the cache controller 16. When the CPU 12 specifies the resource ID of an I/O or DRAM and a data address, and issues access protocols such as load command, store command, or prefetch command, the scheduler 22 of the
10 bridge module 20 registers the access protocol slot in the buffer memory 24. The scheduler 22 controls static resource data such as the access waiting time and access speed of each resource, and dynamic data such as the wait state of the registered access protocols. Based on these data, timing for access protocol issuance is scheduled and access protocol is
15 set up in the request queue of each resource.

The I/O controller 26 and the memory controller 28 read out the access protocols respectively set up in their request queue, and respectively access the I/O module 30 and the
20 memory module 40 to read out the specified address data. The readout data is delivered to the cache controller 16 of the processor module 10 via the internal bus 32. The cache controller 16 caches said data in the cache memory 14, and at the same time provides the CPU 12 with said data. Then, when the CPU 12 accesses the data with the same address, the cache
25 controller 16 delivers the cached data from the cache memory 14 to the CPU 12. In the case of a cache miss, the data is

read out as described above via the bridge module 20.

The processor module 10 with the bridge module 20 is an example of a cache memory controller according to the present invention, and the scheduler 22 with the buffer memory 24 of
5 the bridge module 20 is an example of a prefetch controller according to the present invention.

The CPU 12 executes the compiler loaded in the memory module 40 and compiles the source code of the program and generates the object code. In the generation process of this
10 object code, a prefetch command is inserted.

Fig. 2 is a flowchart describing the program compiling process. The compiler analyzes the source code of the program and converts it into an intermediate code (S10). Then, the compiler determines the data to be prefetched among
15 the data accessed by the load command based on the intermediate code, and calculates the address of that data (S12). Then, the compiler inserts the prefetch command, in which the address of that data is specified as an argument, into a position before the execution sequence position of the
20 load command, and adds the information specifying the data utilization time as another argument in the prefetch command, which shows how many steps from the prefetch command insert position the actual data utilization time will be (S14).
Thus, the compiler optimizes the intermediate code into which
25 the prefetch command with the information specifying the data utilization time (hereinafter referred to as "extended

prefetch command") is inserted (S16), and generates an object code (S18). The extended prefetch command can also be inserted by a carefully tuned assembler program, instead of the compiler.

5 Fig. 3(a) through Fig. 3(d) are diagrams describing examples of programs into which an extended prefetch command has been inserted according to the compiling process in Fig. 2. The program 60 includes access protocols such as the load command (LD), the store command (ST), and the extended
10 prefetch command (PreFetch). In the example in Fig. 3(a), the extended prefetch command 62a to pre-cache the data area *yyy*, which includes the data *yyy-0* and *yyy-1* for use in the load commands 64 and 66, is inserted into a position *n* steps before the execution sequence position of the load command
15 64.

 For the arguments of the extended prefetch command 62a, the data area *yyy* to be prefetched, and the utilization start time *n*, are specified. With the utilization start time *n*, it is possible to know that access to the data area *yyy* starts
20 at a position *n* steps from the insert position of the extended prefetch command 62a. However, it is not known whether or not there are accesses to the data area *yyy* at and/or after the position (*n+1*) steps from the said insert position. For normally arrayed data the consecutive data
25 within the data area *yyy* are accessed simultaneously so that there is the possibility that data within the data area *yyy*

may be used at and/or after the position $(n+1)$ steps from said insert position. Therefore, what can only be judged from the utilization start time n of the extended prefetch command 62a is that there will be some access to the data area yyy at and/or after the position n steps from said insert position, as marked with diagonal lines.

In the example in Fig. 3(b), for the arguments of the extended prefetch command 62b, as information specifying the utilization time, the utilization start time n and the utilization finish time m of the data area yyy to be prefetched, are specified. The data area yyy is accessed for the first time by the load command 64, n steps after the insert position of the extended prefetch command 62b, and is accessed for the last time by the store command 68, m steps after said insert position. In other words, only in the section marked with diagonal lines will the data area yyy be accessed. Thus, by specifying both the utilization start time n and the utilization finish time m of the data area yyy it is possible to define the period during which the cache of the data area yyy should be locked.

In the example in Fig. 3(c), for the arguments of the extended prefetch command 62c, the utilization start time n of the data area yyy to be prefetched, and the priority c , are specified. The priority c is used to decide which prefetch command should be issued first when multiple prefetch commands are registered. The priority c is also

used to decide which cache data should be cached out first when the prefetched data is cached out.

In the example in Fig. 3(d), for the arguments of the extended prefetch command 62b, as information specifying the utilization time of the data area *yyy* to be prefetched, the
5 utilization start time *n*, and the utilization finish time *m*, which were previously described, are specified, and in addition the priority *c* is also specified.

Fig. 4 is a diagram describing how the cache of the
10 cache memory 14 is locked by issuance of the extended prefetch command. Here, an explanation is given using the extended prefetch command 62d in Fig. 3(d) as an example. When the program 60 is executed by the CPU 12, and the program counter PC reaches the position as shown in the
15 diagram, the prefetch command 62d is issued. Using the following parameters in the prefetch command: the data area *yyy*; the utilization start time *n*; the utilization finish time *m*; the priority *c*; and the program counter PC value *pc*, the CPU 12 issues the just-in-time protocol LoadtoCache (*yyy*,
20 *pc*, *n*, *m*, *c*) to the scheduler 22.

The scheduler 22, which received a just-in-time protocol issued by the CPU 12, gives instruction to the cache controller 16 to make a reservation to lock the cache of the data area *yyy* up to (*pc+m*) steps from the current program
25 counter PC value *pc*. Based on the static characteristics such as the latency and bandwidth of the resources like DRAM

and I/O to be accessed, and the dynamic condition of the request queues of the resources, the scheduler 22 schedules the issuance timing of the prefetch command 62d and sets up the actual protocol in the request protocol queues of the resources. At the point when the actual protocol is issued, the scheduler 22 gives instruction to start locking the cache of the data area *yyy* to the cache controller 16. The cache controller 16 locks the cache of the data area *yyy* during the period from the point when the instruction to start locking is received to the point when locking is finished, which is specified by the instruction making a reservation.

Fig. 5 is a block diagram showing the bridge module 20, which was described in Fig. 1, in terms of its functions. The protocol slot generation unit 50, the cost evaluation unit 52, and the protocol issuance unit 54, are included in the configuration of functions in the scheduler 22 in Fig. 1. The schedule list storage unit 56 and the regular protocol queue storage unit 58 are included in the configuration of functions in the buffer memory 24 in Fig. 1.

The protocol slot generation unit 50 receives access protocols from the CPU 12. For access protocols, there is the just-in-time protocol based on the extended prefetch command, and the regular access protocol based on the regular load command or store command.

Fig. 6 is a flowchart describing the protocol slot generation process by the protocol slot generation unit 50.

First, the protocol slot generation unit 50 identifies the type of access protocol received from the CPU 12, whether a just-in-time protocol or other (S20). If it is a just-in-time protocol (Y of S20), the utilization time, which was
5 specified to be n steps from the program counter PC value pc by the extended prefetch command, is corrected based on the unit time of the counter in the scheduler 22 (S22). Next, the protocol slot is generated, which shows the time when the resource will be occupied by the access based on this just-
10 in-time protocol (S24). The generated protocol slot is registered in the schedule list 70 of the just-in-time protocol stored in the schedule list storage unit 56 (S26).

In Step S20, when the access protocol received from the CPU 12 is not a just-in-time protocol (N of S20), the
15 protocol slot generation unit 50 processes this access protocol as a regular protocol and generates the protocol slot indicating the resource occupation time (S28). The generated protocol slot is inserted into the regular protocol queue 90 stored in the regular protocol queue storage unit 58
20 (S30).

Fig. 7 is a diagram describing the schedule list 70 of the just-in-time protocol stored in the schedule list storage unit 56 in Fig. 5. Setting the time, which has been
corrected based on the unit time of the counter of the
25 scheduler 22, on the vertical axis, the registration state at the present time st of the protocol slot of the schedule list

70 is diagrammatically shown. The schedule list 70 is divided into the minimum units of time for the counter of the scheduler 22. As marked with diagonal lines, the slots 72a, 74, and 76 of the just-in-time protocol are respectively
5 stored after a period of 2 units of time, 4 units of time, and 8 units of time, from the present time st . This means that the utilization times of the data prefetched by these protocol slots 72a, 74, and 76 are respectively $(st+2)$, $(st+4)$, and $(st+8)$. However, this diagram is only a pattern
10 diagram to facilitate understanding by showing the storage positions of the protocol slots in accordance with the utilization start time specified by each protocol slot. In actual implementation, the protocol slots are registered in the schedule list 70, with a combination of the utilization
15 time sct_i and the priority c_i as an index, and they are sorted in an order starting from the earliest utilization time sct_i to form a list structure connected as shown with the pointers in the diagram.

A just-in-time protocol is generally written as jtp
20 $(sct, r_{from}, pt_{from}, r_{to}, pt_{to}, c)$. Here, the first parameter sct is the utilization time and the sixth parameter c is the priority. The second parameter r_{from} is the data transfer resource and the fourth parameter r_{to} is the data receiving resource. The third parameter pt_{from} is the time that the data
25 transfer resource is occupied and the fifth parameter pt_{to} is the time that the data receiving resource is occupied. The

resource occupation time is determined based on the data transfer speed and latency of the resource etc. For example, for resources, there are DRAM, I/O, cache memory etc. Data transfer from I/O to cache memory or data transfer from cache memory to I/O can be described using this description method.

In the example of Fig. 7, there are four kinds of resources $r1$ through $r4$ to be accessed. The just-in-time protocol jtp_i is an access protocol to transfer data by occupying the data transfer resource $r3$ and the data receiving resource $r1$, respectively, for the occupation time $pt3$ and $pt1$. This protocol can be written as $jtp_i (sct_i, r3, pt3, r1, pt1, c_i)$. As shown in the diagram, the data structure 72b of the protocol slot 72a of the just-in-time protocol jtp_i is to specify the occupation time $pt1$ of the first resource $r1$ and the occupation time $pt3$ of the third resource $r3$.

Here, within the four resources $r1$ through $r4$, as for the first resource $r1$ and the second resource $r2$, their occupation time is counted based on the minimum time unit of the counter of the scheduler 22. Meanwhile, the occupation time of the third resource $r3$ is counted based on a double-length time unit and the occupation time of the fourth resource $r4$ is counted based on a quadruple-length time unit. This is because the minimum unit for an access occupation time varies with the resource. In the following, to make the description simple, the occupation time will be expressed

based on the minimum unit of the counter of the scheduler 22
regardless of what the minimum unit of the access occupation
time for each resource is. For example, in the just-in-time
protocol jtp_i , the third resource $r3$ is occupied for just 1
5 unit of time based on the minimum unit of access occupation
time of the third resource $r3$. However, this corresponds to
3 units of time in terms of the minimum unit of the counter
of the scheduler 22. So, for the sake of convenience, the
occupation time $pt3$ of the third resource $r3$ is expressed
10 based on the minimum unit of the counter of the scheduler 22,
which comes out to be 3 units of time. Therefore, the just-
in-time protocol jtp_i can be written as $jtp_i (sct_i, r3, 3, r1,$
 $2, c_i)$. Here, the occupation start time of the data
receiving resource $r1$ is 1 unit later than the occupation
15 start time of the transfer resource $r3$. This means that the
data transfer to the data receiving resource $r1$ does not
start until 1 unit of time has passed after the start of the
access of the data transfer resource $r3$.

Fig. 8 is a diagram describing the regular protocol
20 queue 90 stored in the regular protocol queue storage unit 58
in Fig. 5. The regular protocol is issued from the CPU 12 by
the actual data access timing, which is different from the
just-in-time protocol issued from the CPU 12 in advance of
the data access. The protocol slot generation unit 50
25 inserts the regular protocol slots one after another in order
starting from the first one sent from the CPU 12 to the last

in the regular protocol queue 90. However, the protocol slot generation unit 50 can optimize the protocol slots registered in the regular protocol queue 90 by changing the order so that there is no vacant time, or change the sequence based on
5 priority.

The regular protocol can be generally written as np (r_{from} , pt_{from} , r_{to} , pt_{to} , c). The definition of parameters is the same as the case for the just-in-time protocol. The priority c , which is the fifth parameter, is an option and it
10 may not be used in some cases. The utilization method of the priority c is the same as the case for the just-in-time protocol. In the example in Fig. 8, at the present time st , the slots of the regular protocols from i th to $(i+3)$ th, which are np_i , np_{i+1} , np_{i+2} , and np_{i+3} , are registered in the regular
15 protocol queue 90. The regular protocols from i th to $(i+3)$ th can be written respectively as np_i ($r1$, 6 , $r2$, 6 , c_i), np_{i+1} ($r3$, 3 , $r4$, 6 , c_{i+1}), np_{i+2} ($r2$, 2 , $r3$, 3 , c_{i+2}), and np_{i+3} ($r4$, 6 , $r3$, 3 , c_{i+3}). The regular protocols in the output slot located at the head of the regular protocol queue 90 are
20 taken out in sequence and issued.

With reference again to the functional configuration diagram in Fig. 5, the cost evaluation unit 52 evaluates the issuance cost of the regular protocol in a wait status in the regular protocol queue 90, and the issuance cost of the just-
25 in-time protocol registered in the schedule list 70, and compares these issuance costs. The protocol issuance unit 54

issues the protocol with the smaller issuance cost. Based on the resource ID, the issued protocol is sent to the I/O controller 26 if it is a command for I/O, and to the memory controller 28 if it is a command for a main memory like DRAM, and executed.

Fig. 9 is a flowchart showing the evaluation process for the issuance cost of access protocol by the cost evaluation unit 52. Fig. 10 is a diagram describing the schedule list 70, and the registration state of the protocol slots in the regular protocol queue 90. Explanation will be given with reference to these diagrams.

The cost evaluation unit 52 calculates the regular protocol issuance cost N_p (S40). The regular protocol issuance cost N_p is generally a function of the regular protocol np in the output slot of the regular protocol queue 90 and the present time st . Here, as shown in the formula below, when a regular protocol np in the output slot is issued, the cost N_p can be obtained by calculating the linear sum of the cost $RC_{from}(t)$ due to the data transfer resource r_{from} being occupied at the time t , and the cost $RC_{to}(t)$ due to the data receiving resource r_{to} being occupied at the time t .

$$N_p = np_cost_func(np, st) = RC_{from}(st) + RC_{to}(st+\alpha)$$

Here, the constant α is the time difference between the occupation start time of the data transfer resource and the occupation start time of the data receiving resource.

In the example of the regular protocol queue 90 in Fig. 10, the regular protocol $np1$ exists in the output slot. If this regular protocol $np1$ is issued at the present time st , access to the data transfer resource $r3$ starts at the present
5 time st , and the data transfer resource $r3$ is occupied for just 3 units of time. The occupation cost for this time is calculated by $RC_3(st)$. Meanwhile, access to the data receiving resource $r2$ starts 2 units of time later, and the data receiving resource $r2$ is occupied for just 1 unit of
10 time. The occupation cost for this time is calculated by $RC_2(st+2)$. Therefore, the regular protocol issuance cost Np is calculated by the formula: $Np = RC_3(t) + RC_2(t+2)$.

Next, the cost evaluation unit 52 calculates the just-in-time protocol issuance cost $jtpcost$ (S42). The cost
15 evaluation unit 52 does the evaluation of the issuance cost $jtpcost$ in regard to all the protocol slots registered within the schedule list 70. However, the issuance cost $jtpcost$ may be evaluated for only the protocol slot with the utilization time sct , which is the closest to the present time st , or the
20 protocol slot with the highest priority c . Also, the issuance cost $jtpcost$ may be evaluated for several protocol slots in an order starting from the utilization time sct , which is the closest to the present time st , or for several protocol slots in an order starting from the protocol slot
25 with the highest priority c . In the example in Fig. 10, in the schedule list 70 the slots for two just-in-time protocols

jtp1 (*sct1*, *r1*, *pt1*, *r2*, *pt2*, *c1*) and *jtp2* (*sct2*, *r2*, *pt2*, *r4*, *pt4*, *c2*) are registered. The cost evaluation unit 52 individually calculates the issuance costs *jtpcost1* and *jtpcost2* for the respective protocol slots.

5 The just-in-time protocol issuance cost *jtpcost* is calculated based on a combination of several cost factors. The first cost factor is the regular protocol issuance lag cost *LC*. This is the cost per unit of time accrued due to an issuance lag of the regular protocol, which is in a wait
10 status in the regular protocol queue 90 when the just-in-time protocol is issued. This regular protocol issuance lag cost *LC* is generally a function applied to all the regular protocols stored in the regular protocol queue 90. Here, it is the linear sum of the lag costs $LCs(np_i)$ calculated for
15 the respective regular protocols np_i . In the example of Fig. 10, at the present time *st* five regular protocols, *np1* through *np5*, are in a wait status in the regular protocol queue 90, therefore, the regular protocol issuance lag cost *LC* is calculated by the following formula:

$$20 \quad LC = Lcfunc(np1, np2, np3, np4, np5) = LC(np1) + LC(np2) + LC(np3) + LC(np4) + LC(np5)$$

Here, the lag cost $LC(np_i)$ is the cost accrued when the issuance of the regular protocol np_i has a lag time of just 1 unit of time.

25 When the priority c_i is specified in the regular protocol np_i the lag cost is written as $LC(np_i, c_i)$, and its

value may be set so that it reflects the priority c_i for each regular protocol np_i . For example, the lag cost $LC(np_i, c_i)$ may be evaluated higher for the regular protocol np_i with a high priority c_i .

5 In the example in Fig. 10, when the first just-in-time protocol $jtp1(sct1, r1, pt1, r2, pt2, c1)$ is issued, as shown in the slot 82 of this protocol, the data receiving resource $r2$ is occupied for 5 units of time from the present time st , which overlaps the time for which the resource $r2$ is occupied
10 by the regular protocol $np1$ in the output slot of the regular protocol queue 90. Therefore, to execute the just-in-time protocol $jtp1$, as shown in the figure, the output slot 86 of the regular protocol queue 90 must be shifted backward by
15 $ltc1$ time, in other words 3 units of time. By having the regular protocol $np1$ in the output slot 86 delay for 3 units of time, other regular protocols $np2$ through $np5$, which are in a wait status in the regular protocol queue 90, also delay for 3 units of time as a whole. Therefore, as a whole, by issuance of the just-in-time protocol $jtp1$, the cost $LC *$
20 $ltc1$, which is the regular protocol issuance lag cost LC multiplied by the lag time $ltc1$ of the output slot 86, accrues.

In the same way, when the second just-in-time protocol $jtp2$ is issued, the lag time $ltc2$ can be calculated based on
25 the competing relationship over the resource occupation time between the slot 84 of this protocol and the protocol slots

within the regular protocol queue 90. In general, the regular protocol issuance lag cost accrued from the issuance of the just-in-time protocol jtp_i is calculated by the formula: $LC * ltc_i$.

5 The second cost factor of the just-in-time protocol issuance cost $jtpcost$ is the cache retention cost $CacheRC$ of the prefetched data. This is the cost per unit of time, which accrues due to the prefetched data staying in the cache memory 14 when the just-in-time protocol is issued. The
10 value $ct_i = (sct_i - st)$, which is the present time st subtracted from the utilization time sct_i of the data to be prefetched by the just-in-time protocol jtp_i , equals the time that the prefetched data stays in the cache memory 14 when the just-in-time protocol jtp_i is issued at the present time
15 st . Therefore, the cache retention cost accrued due to the issuance of the just-in-time protocol jtp_i at the present time st is calculated by $CacheRC * (sct_i - st)$: the cache retention cost $CacheRC$ per unit of time multiplied by the retention time $ct_i = (sct_i - st)$.

20 The third cost factor is the risk that the data being accessed is not prefetched by the specified utilization time due to an issuance delay of the just-in-time protocol jtp_i . With the just-in-time protocol jtp_i and the present time st being the parameters of a function, it is calculated by
25 $Time_Limit_Func(jtp_i, st)$. As an example, this risk is defined as shown in the formula below:

$Time_Limit_Func(jtp_i, st) = \{Time_Limit \text{ when } sct_i - st < \max(pt_{from}, pt_{to}) + a, \text{ or otherwise } 0\}.$

When the prefetched data cache retention time ($sct_i - st$) is less than the greatest value $\max(pt_{from}, pt_{to})$ of either
 5 the occupation time pt_{from} of the data transfer resource r_{from} and the occupation time pt_{to} of the data receiving resource r_{to} resulting from prefetching, data prefetching cannot be completed by the utilization time sct_i . So, the cache retention time ($sct_i - st$) is compared to the value
 10 calculated by adding the constant a , which is given to allow some room, to $\max(pt_{from}, pt_{to})$. When the cache retention time ($sct_i - st$) is less, the risk value shall be the positive constant $Time_Limit$, and when the cache retention time ($sct_i - st$) is greater, the risk value shall be 0.

15 The just-in-time protocol jtp_i issuance cost $jtpcost_i$ at the present time st is calculated based on the above three cost factors as follows:

$$jtpcost_i = jtp_cost_func(st, jtp_i) \\ = LC * ltc_i + (sct_i - st) * CacheRC - Time_Limit_Func(jtp_i, st)$$

20

The third term $Time_Limit_Func(jtp_i, st)$ takes a positive value when a situation draws near in which prefetching cannot be completed, and its value is subtracted from the sum of the first term $LC * ltc_i$ and the second term
 25 $(sct_i - st) * CacheRC$, which results in lowering the value of $jtpcost_i$ as a whole. Therefore, in a situation in which it

appears that keeping of the just-in-time protocol will be difficult, a balance is struck between the cost accrued by executing the just-in-time protocol and the risk carried by not executing the just-in-time protocol.

5 Going back to Fig. 9, when the just-in-time protocol issuance cost $jtpcost$ is smaller than the regular protocol issuance cost Np at the present time st (Y of S44), the protocol issuance unit 54 issues a just-in-time protocol (S46), and if the just-in-time protocol issuance cost $jtpcost$
10 is not smaller (N of S44), the protocol issuance unit 54 issues a regular protocol (S48). Once one of the two protocols is issued, the process goes back to Step S40. Then, a new cost calculation is done at that time point and based on the cost evaluation at each time point an
15 appropriate protocol is selected and issued.

 In the example of Fig. 10, the first just-in-time protocol $jtp1$ issuance cost $jtpcost1$, and the second just-in-time protocol $jtp2$ issuance cost $jtpcost2$, which are registered in the schedule list 70, and the issuance cost Np
20 of the regular protocol $np1$ in the output slot of the regular protocol queue 90, are compared, and the smallest cost protocol is issued by the protocol issuance unit 54. Here, it must be noted that the just-in-time protocol issuance cost needs to be re-calculated for each time because the wait
25 state of a protocol within the regular protocol queue 90 varies each time and accordingly the value of the first term,

$LC * ltc_i$ of the issuance cost calculation formula, changes. Also, there is the possibility of the values associated with the relation between the first just-in-time protocol $jtp1$ issuance cost $jtpcost1$ and the second just-in-time protocol
 5 $jtp2$ issuance cost $jtpcost2$ changing each time due to the same reason above. So, there is no guarantee that the just-in-time protocol with a utilization time sct , which is closer to the present time st , will have a smaller issuance cost.

In addition, the just-in-time protocol issuance cost
 10 $jtpcost_i$ may be evaluated by only partially using the three cost factors. For example, the evaluation can be made by the following formula using only Cache retention cost:

$$jtpcost_i = (sct_i - st) * CacheRC$$

Also, the following formula can be used for evaluation, which
 15 also includes use of the risk factor that prefetch cannot be completed:

$$jtpcost_i = (sct_i - st) * CacheRC - Time_Limit_Func(jtp_i, st)$$

When using the above formulae, the just-in-time protocol issuance cost can be evaluated independently of the condition
 20 of the protocol slot of the regular protocol queue 90, which simplifies cost calculation.

As has been described so far, according to the embodiments, by specifying a utilization time of data to be prefetched in the prefetch command, it is possible to predict
 25 a period of time for which the prefetched data stays in the cache, which enables decision making about the issuance

timing of the prefetch command through evaluating the cache retention cost. Further, by making a cost evaluation in consideration of the competing relationship between the resource occupation times, it is possible to issue a prefetch
5 command while striking a balance with other access command executions. As a result, the prefetch command is not executed needlessly but according to a plan and the cache usability is improved.

Although the present invention has been described by
10 way of exemplary embodiments, it should be understood that many changes and substitutions may be made by those skilled in the art without departing from the scope of the present invention which is defined by the appended claims. Some such alterations are stated as follows.

15 In the description above, in the extended prefetch command, the utilization start point and finish point of the prefetched data were specified based on the relative number of steps counted from the insert position of the extended prefetch command. However, the utilization start point and
20 finish point of the prefetched data may be specified based on an absolute time.

Further, in the description above, the issuance costs for both the just-in-time protocol and regular protocol are calculated and compared to decide which access protocol shall
25 be issued. However, it is also acceptable to simply calculate the just-in-time protocol issuance cost and if its

value is equal to or below a specified value the decision to issue a just-in-time protocol may be made.

CLAIMS

1. A prefetch command control method comprising generating a prefetch command for data with information specifying a utilization time of the data, and scheduling an issuance timing of the prefetch command according to the information specifying the utilization time.
2. The method of Claim 1, further comprising calculating an issuance cost of the prefetch command based on a period of time for which a prefetched data stays in a cache when the prefetch command is issued, and wherein the issuance timing of the prefetch command is scheduled according to the issuance cost.
3. The method of Claim 2, further calculating a cost, which accrues due to an issuance lag of a regular access command for data when the prefetch command is issued, to evaluate the issuance cost of the prefetch command.
4. The method of any one of Claims 2-3, further evaluating a risk depending on a time constraint of prefetching defined by a period of time from a current time to the specified utilization time, to evaluate the issuance cost of the prefetch command.

5. The method of Claim 2, further comprising calculating an issuance cost of a regular access command for data based on a period of time for which a resource is occupied when the regular access command is issued, and comparing the issuance cost of the prefetch command with the issuance cost of the regular access command to decide which of the prefetch command or the regular access command shall be issued.

6. A prefetch command control apparatus comprising:

a storage unit which stores a prefetch command for an external resource with information specifying data utilization time added;

a cost evaluation unit which evaluates an issuance cost which accrues due to a prefetched data staying in a cache by the specified utilization time when the stored prefetch command is issued; and

an issuance unit which adjusts a timing when the stored prefetched command is issued for the external resource according to an evaluation result by the cost evaluation unit.

7. The apparatus of Claim 6, wherein the cost evaluation unit compares the issuance cost of the prefetch command with an issuance cost of a regular access command stored in an access request queue for the external resource, and the issuance unit decides which of the prefetch command and the regular access command shall be issued at a current time

according to a comparison result by the cost evaluation unit.

8. The apparatus of Claim 6, wherein the cost evaluation unit calculates a cost, which accrues due to an issuance lag of a regular access command stored in an access request queue for the external resource when the prefetch command is issued, to evaluate the issuance cost of the prefetch command.

9. The apparatus of any one of Claims 6-8, wherein the cost evaluation unit evaluates a risk depending on a time constraint of prefetching defined by a period of time from a current time to the specified utilization time, to evaluate the issuance cost of the prefetch command.

10. A cache memory control apparatus comprising:

a cache memory which caches data obtained from an external resource and provides the cached data to a processor;

a scheduler which schedules an access protocol for the external resource received from the processor and sets up the scheduled access protocol in a request queue for the external resource; and

an external resource controller which reads the access protocol from the request queue and accesses the external resource to provide data to the cache memory, and

wherein the scheduler comprising:

a storage unit which stores a prefetch command for an external resource with information specifying data utilization time added;

a cost evaluation unit which evaluates an issuance cost which accrues due to a prefetched data staying in a cache by the specified utilization time when the stored prefetch command is issued; and

an issuance unit which adjusts a timing when the stored prefetched command is issued for the external resource according to an evaluation result by the cost evaluation unit.

11. The apparatus of Claim 10, wherein the scheduler locks a cache of the prefetched data in the cache memory according to the information specifying data utilization time after the prefetch command is issued.

12. The apparatus of Claim 10, wherein the cost evaluation unit compares the issuance cost of the prefetch command with an issuance cost of a regular access command stored in an access request queue for the external resource, and the issuance unit decides which of the prefetch command and the regular access command shall be issued at a current time according to a comparison result by the cost evaluation unit.

13. The apparatus of Claim 10, wherein the cost evaluation unit calculates a cost, which accrues due to an issuance lag

of a regular access command stored in an access request queue for the external resource when the prefetch command is issued, to evaluate the issuance cost of the prefetch command.

14. The apparatus of any one of Claims 10-13, wherein the cost evaluation unit evaluates a risk depending on a time constraint of prefetching defined by a period of time from a current time to the specified utilization time, to evaluate the issuance cost of the prefetch command.

15. A computer program executable by a computer, the program comprising:

analyzing a source code and converting the source code into an intermediate code;

determining data to be prefetched based on the intermediate code, and then inserting an access command for the data as a prefetch command into a position before an original execution sequence position, and adding information specifying a utilization time of the data in the prefetch command; and

optimizing the intermediate code into which the prefetch command is inserted and generating an object code.

16. The program of Claim 15, wherein the information specifying the utilization time includes information specifying a utilization start time of the data to be

prefetched, which is defined by a relative position from a position where the prefetch command is inserted.

17. The program of Claim 16, wherein the information specifying the utilization time further includes information specifying a utilization finish time of the data to be prefetched, which is defined by a relative position from a position where the prefetch command is inserted.

FIG. 1

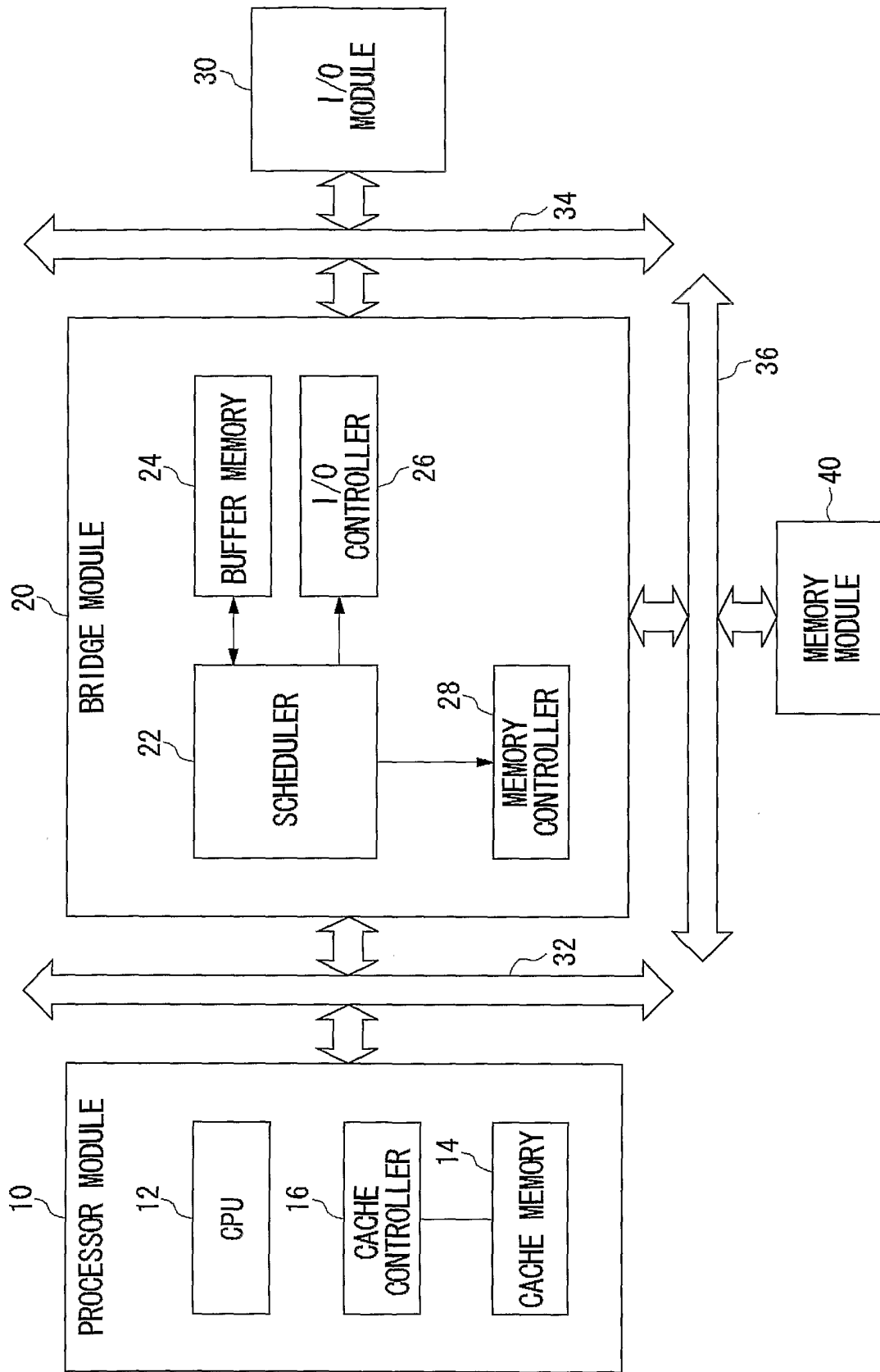


FIG.2

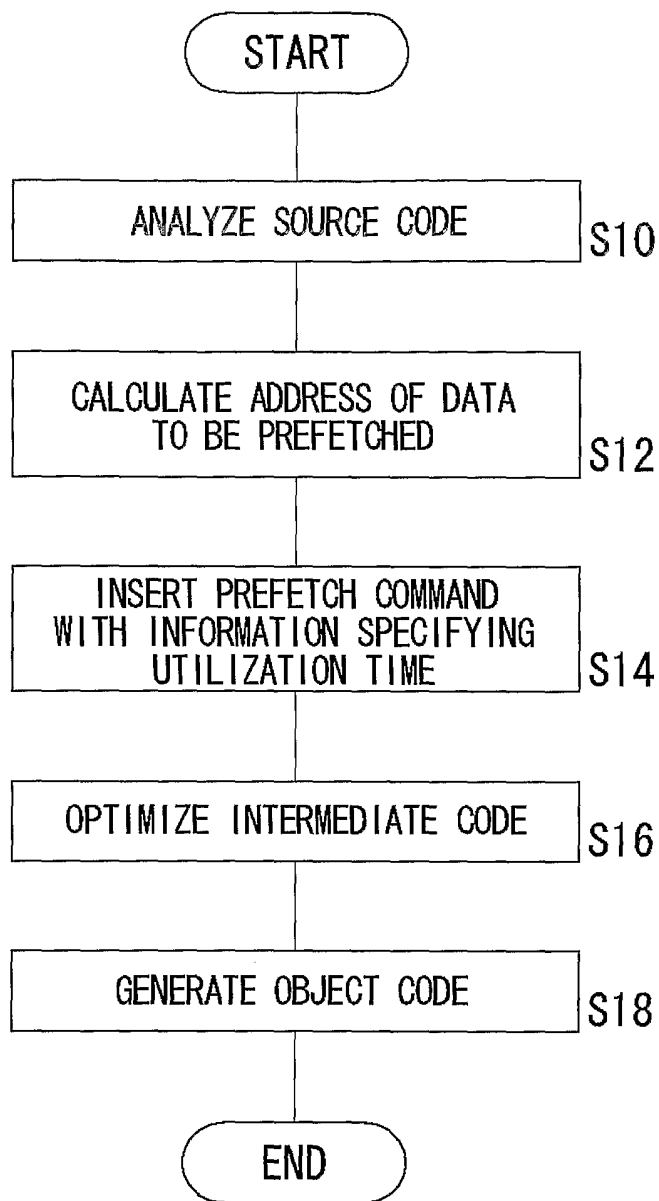


FIG.4

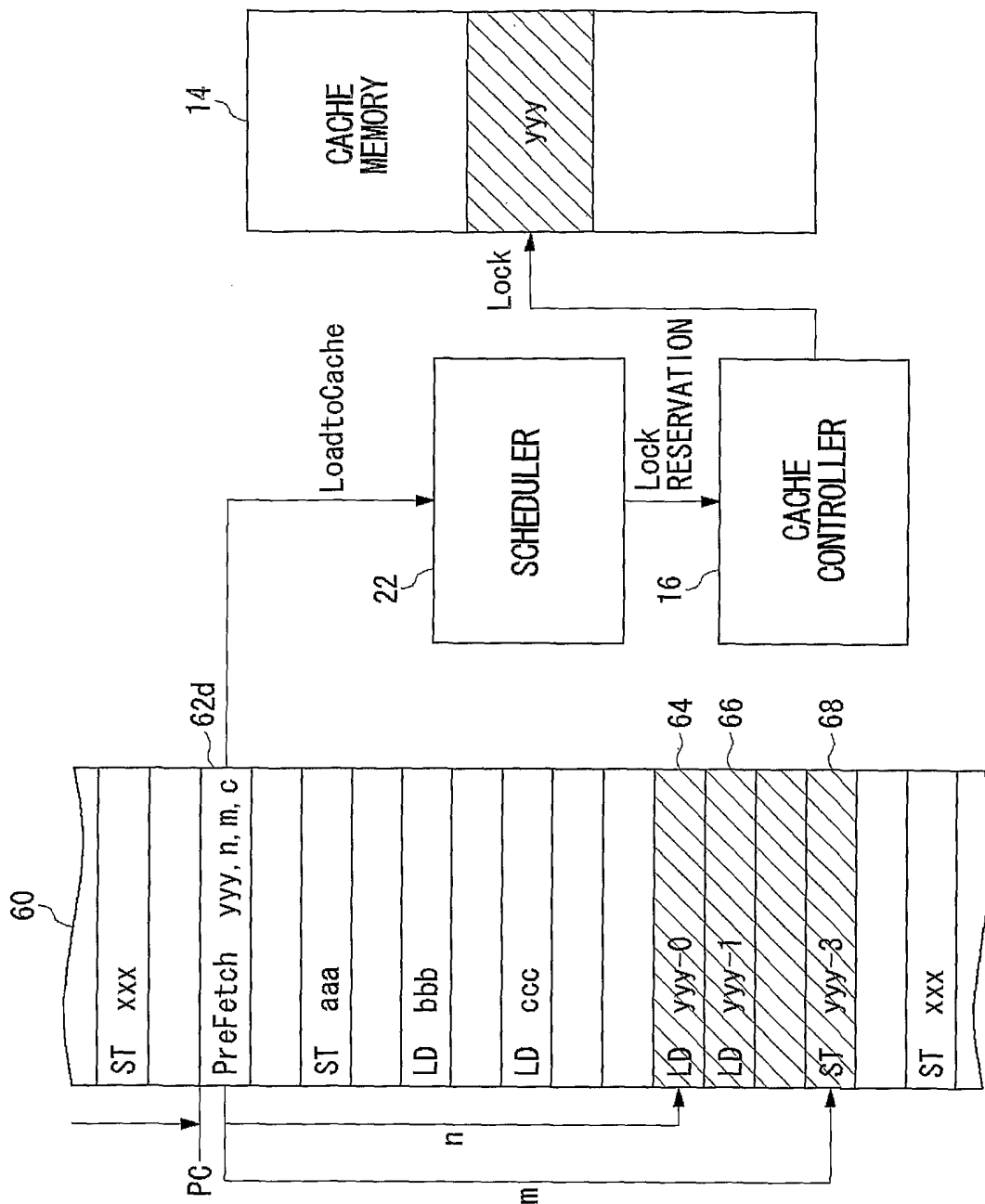


FIG.5

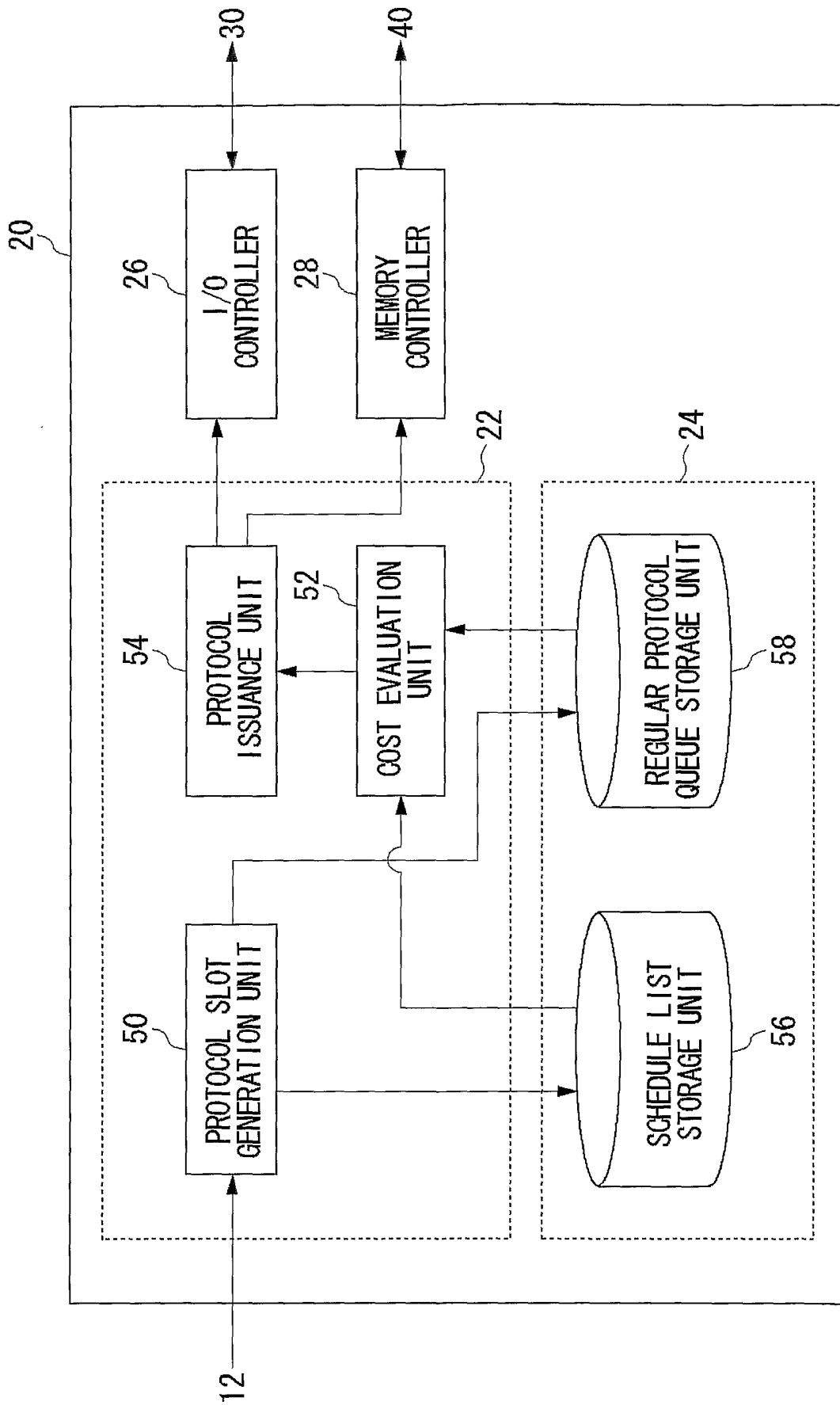


FIG.6

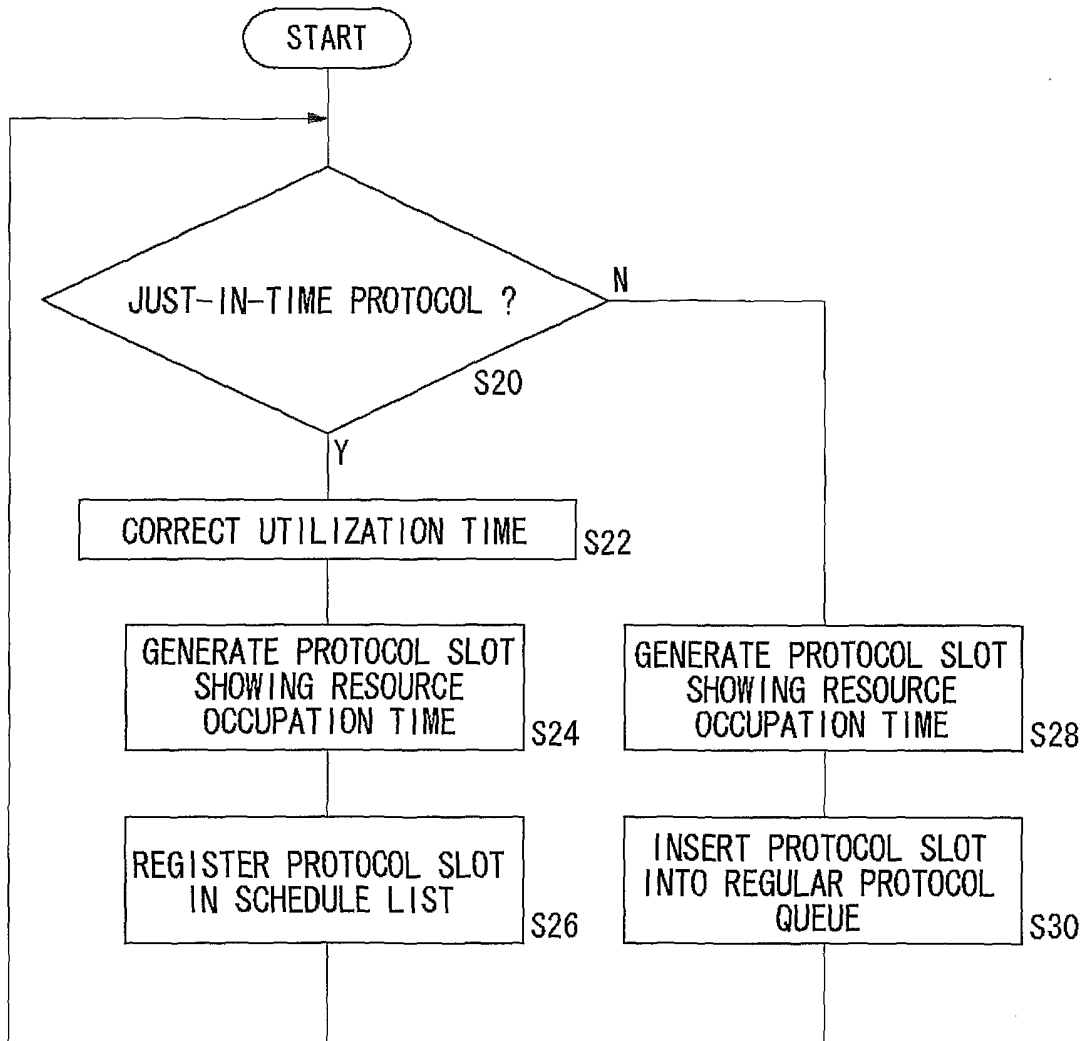


FIG.7

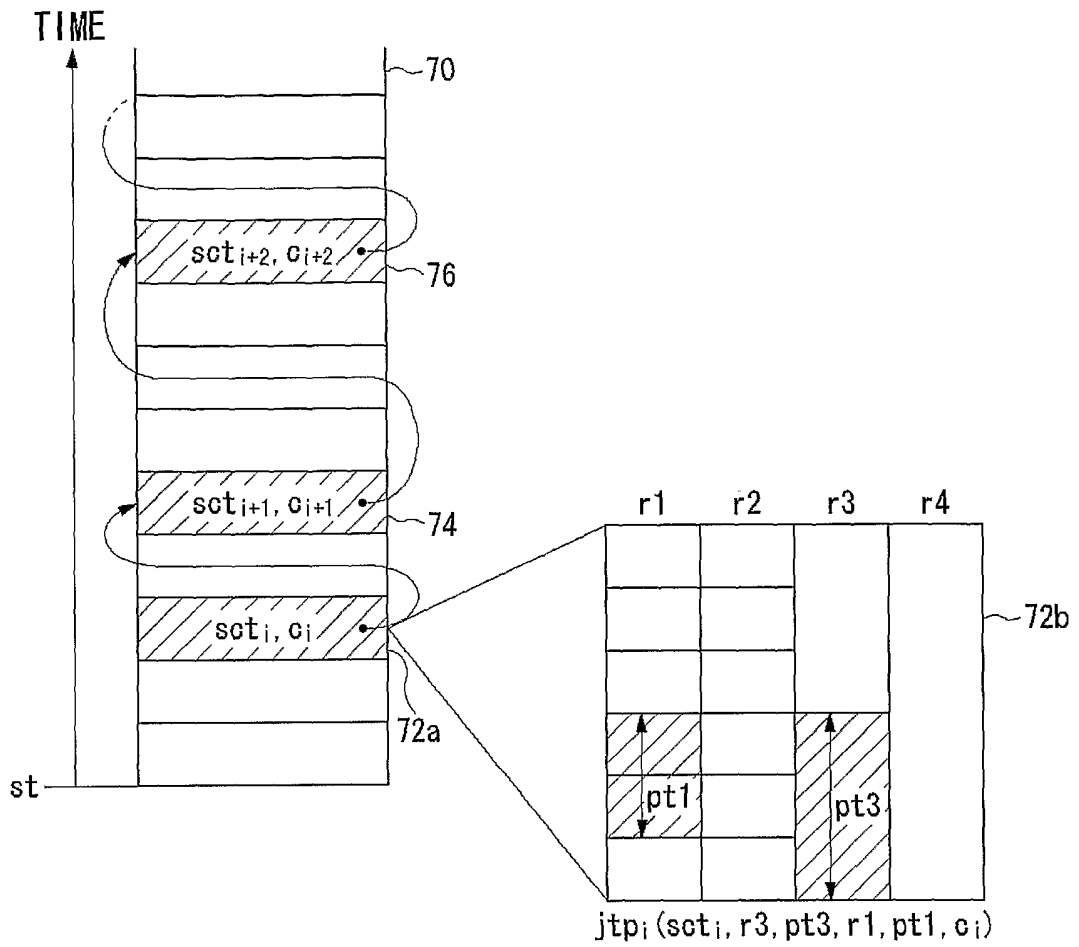


FIG.8

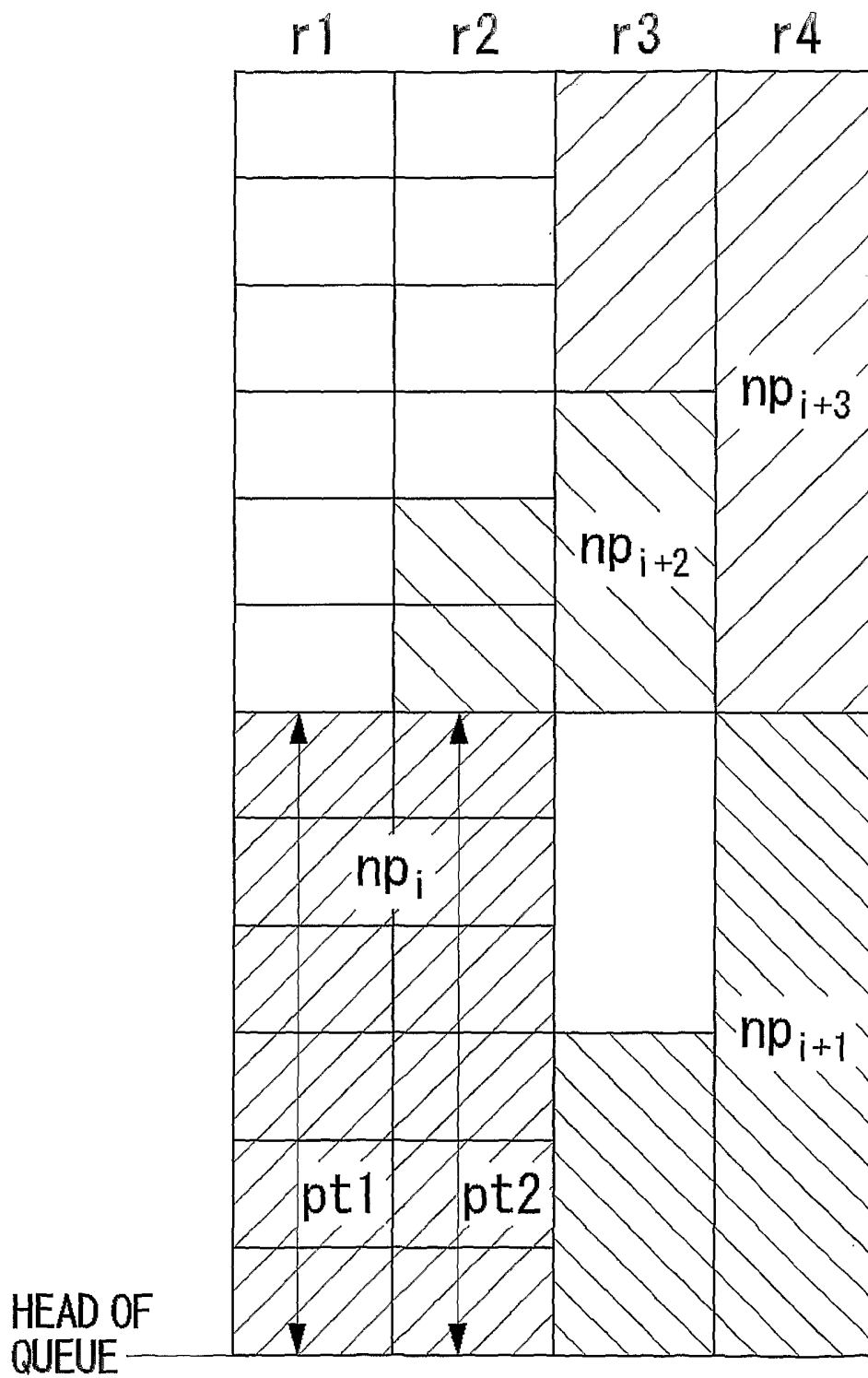


FIG.9

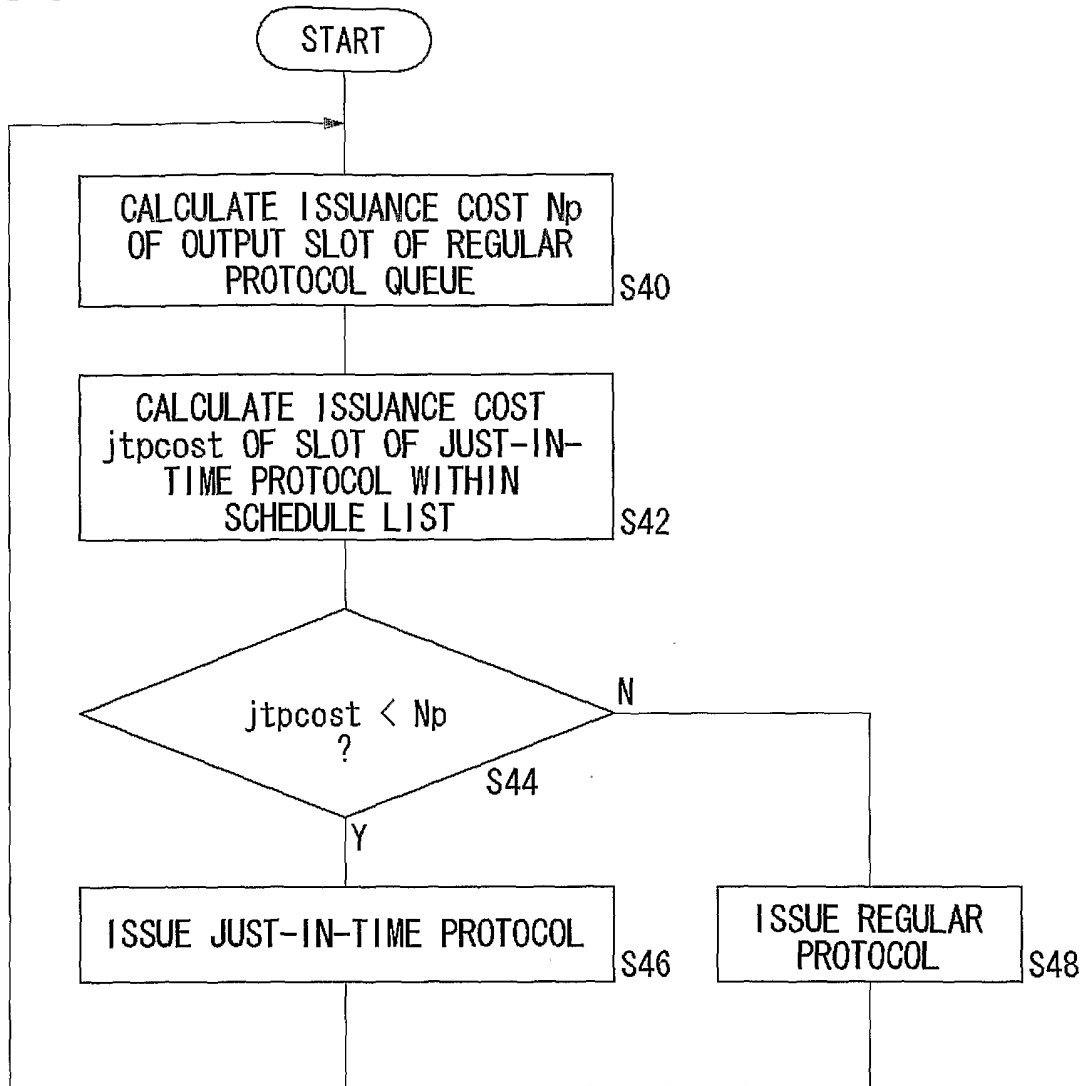


FIG. 10

