INSTRUCTION CACHE 16

TAG ARRAY 26
(n) TAGO

I CLASS ARRAY 27
(n) I CLASS

PRODUCTIVE
ANNOTATION ARRAY 28
(m) BRPDS [(k) NFAPDS

INSTRUCTION
ARRAY 24
(n) INSTRUCTIONS

341
342
34x

SELECTION LOGIC 30
TAG
ICLASS
NFAPD
BRPD
INSTR

N instruction class (IClass) fields, m branch prediction (BRPD) and k next fetch address fields are added to each instruction set of n instructions of a cache line of an instruction cache, where m and k are less than or equal to n. The BRPD and NFAPD fields of a cache line are initialized in accordance to a pre-established initialization policy of a branch and next fetch address prediction algorithm while the cache line is first brought into the instruction cache. The sets of IClasses, BRPDS, and NFAPDS of a cache line are accessed concurrently with the corresponding sets of instructions of the cache line. One BRPD and one NFAPD is selected from the set of BRPDS and NFAPDS corresponding to the selected set of instructions. The selected BRPD and NFAPD are updated in accordance to a pre-established update policy of the branch and next fetch address prediction algorithm when the actual branch direction and next fetch address are resolved. Additionally, in one embodiment, m and k are equal to 1, and the selected NFAPD is stored immediately into the NFA register of the instruction prefetch and dispatch unit, allowing the selected NFAPD to be used as the fetch address for the next instruction cache access to achieve zero fetch latency for both control transfer and sequential next fetch.
FIG. 3
START

RECEIVE BRANCH RESOLUTION SIGNAL

BRANCH PREDICTION CORRECT

YES

DETERMINE/RECEIVE NFA RESOLUTION

NEXT FETCH ADDRESS PREDICTION CORRECT

YES

UPDATE SELECTED BRPD (OPTIONAL)

END

FIG. 4B
FIG. 5

START

RECEIVE NEXT FETCH ADDRESS

CACHE HIT?

YES

NO

RETURN INSTRUCTIONS, BRPD, NFAPD

END

INITIATE CACHE FILL

BYPASS INSTRUCTION(S)

ENCODE INSTRUCTION CLASSES

INITIALIZE BRPD'S AND NFAPD'S
### FIG. 6

<table>
<thead>
<tr>
<th>INSTRUCTIONS</th>
<th>BRPD FIELDS 40</th>
<th>NFAPD FIELDS 42</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>0</td>
<td>next seq. addr.</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>next seq. addr.</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>target addr 1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>target addr 2</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>target addr 3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>4</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>34_1</th>
<th>34_2</th>
<th>34_3</th>
<th>34_4</th>
<th>34_5</th>
<th>34_6</th>
<th>34_7</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Next sequence address for NFAPD and BRPD fields.
COMPUTER SYSTEM AND METHOD FOR FETCHING A NEXT INSTRUCTION
CROSS REFERENCE TO RELATED APPLICATIONS

[0001] This is a division of application Ser. No. 08/800, 367, filed Feb. 14, 1997, which is a continuation of application Ser. No. 08/363,107, filed Dec. 22, 1994, which is a continuation application of Ser. No. 07/938,571, filed Aug. 31, 1992, all of which are incorporated herein by reference.

DESCRIPTION OF THE INVENTION

[0002] 1. Field of the Invention

[0003] The present invention relates to the field of computer systems. More specifically, the present invention relates to a computer system having a minimum latency cache which stores instructions decoded to determine class, branch prediction and next fetch address prediction information.

[0004] 2. Background of the Invention

[0005] Historically, when a branch instruction was dispatched in a computer system, instruction fetching and dispatching were stalled until the branch direction and the target address were resolved. Since this approach results in lower system performance, it is rarely used in modern high performance computers. To obtain higher system performance, various techniques have been developed to allow instruction fetching and dispatching to continue in an efficient manner without waiting for the resolution of the branch direction. Central to the efficiency of continuing instruction prefetching and dispatching is the ability to predict the correct branch direction. There are several common approaches to predicting branch direction:

[0006] 1. Static prediction: Under this approach, the higher probability direction for a particular branch instruction is ascertained. When the branch instruction is fetched, the ascertained direction is always taken. For example, a direction for a branch instruction may be set to “Branch Taken”, or alternatively, set to “Branch Not Taken”.

[0007] 2. Dynamic software prediction: Under this approach, a branch prediction algorithm predicts the branch direction.

[0008] 3. Dynamic hardware prediction: Under this approach, a branch prediction algorithm predicts the branch direction based on the branch history information maintained in a branch prediction table.

[0009] The static prediction approach is simple to implement, however, its prediction hit rate is generally less than 75%. Such a prediction hit rate is generally too low for high performance computers. The dynamic software prediction approach generally works quite well when used in conjunction with a compilation technique known as trace scheduling. Without trace scheduling, the prediction hit rate is generally very low. Unfortunately, trace scheduling is difficult to apply to some programs and implementations. The dynamic hardware prediction generally provides an adequate prediction hit rate. However, it increases the complexity of the processor design and requires additional hardware to maintain the separate branch prediction table.

Further, if the size of a cache is enlarged in a redesign, the size of the table would also have to be increased, complicating the redesign process.

SUMMARY OF THE INVENTION

[0010] The present invention relates to a novel computer system. The computer system includes a low latency cache that stores instructions decoded to determine class, branch prediction information, and next address fetch information.

[0011] The present invention includes a cache having a plurality of cache lines. Each cache line includes (n) instructions and (n) instruction class (ICLASS) fields for storing the decoded class information of the instructions respectively. Each cache line also includes one or more branch prediction (BRPD) fields and one or more next fetch address prediction (NFAPD) fields.

[0012] When an instruction is fetched, the corresponding ICLASS field, BRPD field information and the NFAPD information are all provided to the prefetch and dispatch unit of the processor system. The ICLASS information informs the prefetch unit if the fetched instruction is a branch. Since the instruction has already been decoded to determine its class, the need to perform a partial decode in the prefetch and dispatch unit to determine if an instruction is a branch instruction is avoided. If the instruction is a branch instruction, the BRPD field provides a prediction of either “Branch Taken” or “Branch Not Taken”. For non-branch instructions, the BRPD field is ignored. For non-branch instructions, the NFAPD typically contains the next sequential address. For branch instructions, the NFAPD contains either the next sequential address or the target address of the branch instruction. If the BRPD field contains a “Branch Taken” prediction, the corresponding NFAPD field typically contains the target address for the branch instruction. Alternatively, if the BRPD field contains a “Branch Not Taken” status, the corresponding NFAPD field typically contains the next sequential address. In any event, the NFAPD information is used to define the next line from the cache to be fetched, thereby avoiding the need to calculate the next fetch address in the prefetch unit. The prefetch and dispatch unit needs to calculate the next fetch address only when a misprediction of a branch instruction occurs. An update policy is used to correct the BRPD and the NFAPD values in the event the predictions turn out to be wrong.

[0013] The number of BRPD fields and NFAPD fields per cache line varies depending on the specific embodiment of the present invention. In one embodiment, a specific BRPD field and an NFAPD field is provided for each instruction per cache line. If there is more than one branch instruction per cache line, each branch instruction enjoys the benefit of a dedicated branch prediction and next fetch address prediction. In a simplified embodiment one BRPD field and one NFAPD field is shared among all the instructions per cache line. Under these circumstances, only a dominant instruction in the cache line makes use of the BRPD and the NFAPD information. A dominant instruction is defined as the first branch instruction with a “Branch Taken” status in the cache line. For example, with a dominant instruction, the BRPD field is set to “Branch Taken”, and the NFAPD typically contains the target address for the dominant branch instruction. When the instruction is fetched, control is typically transferred to the target address of the dominant instruction.
Since the dominant instruction is the first instruction in a cache line to cause a control transfer, it is not necessary for the other instructions in the cache line to have their own BRPD fields and NFAPD fields respectively.

[0014] The present invention represents a significant improvement over the prior art. The need to perform a partial decode or a next fetch address calculation in the prefetch and dispatch unit is eliminated with a vast majority of die fetched instructions. As such, fetch latency is significantly reduced and processor throughput is greatly enhanced.

BRIEF DESCRIPTION OF THE DRAWINGS

[0015] The objects, features and advantages of the system of the present invention will be apparent from the following detailed description of the invention with references to the drawings in which:

[0016] FIG. 1 is a block diagram of a computer system according to the present invention.

[0017] FIG. 2 illustrates a block diagram of an instruction cache in the computer system of the present invention.

[0018] FIG. 3 illustrates a block diagram of an instruction prefetch and dispatch unit used in the computer system of the present invention.

[0019] FIGS. 4a-4b are two flow diagrams illustrating the operation of the instruction prefetch and dispatch unit.

[0020] FIG. 5 is a flow diagram illustrating the operation of the instruction cache.

[0021] FIG. 6 illustrates exemplary line entries in the instruction cache used in the computer system of the present invention.

DESCRIPTION OF THE EMBODIMENTS

[0022] Referring to FIG. 1, a functional block diagram illustrating a computer system of the present invention is shown. The computer system 10 includes an instruction prefetch and dispatch unit 12, execution units 14, an instruction cache 16, a data cache 18, a memory unit 20 and a memory management unit 22. The instruction cache 16 and data cache 18 are coupled to the instruction prefetch and dispatch unit 12, the execution units 14, and the memory management unit 22 respectively. The prefetch and dispatch unit 12 is coupled to the execution units 14 and the memory management unit 22. The data cache 18 is coupled to memory 20. The instruction cache 16 is coupled to memory 20.

[0023] Cooperatively, the memory management unit 22 and the prefetch and dispatch unit 12 fetch instructions from instruction cache 16 and data from the data cache 18 respectively and dispatch them as needed to the execution units 14. The results of the executed instructions are then stored in the data cache 18 or main memory 20. Except for the instruction prefetch and dispatch unit 12 and the instruction cache 16, the other elements, 14 and 18 through 22, are intended to represent a broad category of these elements found in most computer systems. The components and the basic functions of these elements 14, and 18 through 22, are well known and will not be described further. It will be appreciated that the present invention may be practiced with other computer systems having different architectures. In particular, the present invention may be practiced with a computer system having no memory management unit 22. Furthermore, the present invention may be practiced with a unified instruction/data cache or an instruction cache only.

[0024] Referring now to FIG. 2, a block diagram illustrating the instruction cache 16 of the present invention is shown. The instruction cache 16 includes an instruction array 24, a tag array 26, an ICLASS array 27, a predictive annotation array 28, and selection logic 30. The cache is segmented into a plurality of cache lines 34 through 34. Each cache line 34 includes (n) instructions in the instruction array 24, (m) branch prediction BRPD fields 40, (k) next address prediction NFAPD fields 42 in the predictive annotation array 28, (n) ICLASS fields 44 in the ICLASS array 27, and (n) tags in the tag array 26. It also should be noted that the instruction cache 16 may be set associative. With such in embodiment, individual arrays 24 through 29 are provided for each set in the instruction cache 16.

[0025] Each of the (n) instructions per cache line 34 contained in the instruction cache 16 are decoded to determine their class. In one embodiment, the instruction class encodings are stored in the appropriate ICLASS field 44, when the cache line 34 is being brought into the instruction cache 16. In an alternative embodiment, the instruction class encodings are stored before the cache line 34 is brought into the instruction cache 16. Examples of instruction classes are the program counter (PC) relative branch, register indirect branch, memory access, arithmetic and floating point.

[0026] When the instruction cache 16 receives a next fetch address from the instruction prefetch and dispatch unit 12, the appropriate cache line 34 is accessed. The (n) instructions, the (m) BRPD fields 40, the (k) NFAPD fields 42, the (n) ICLASS fields 44, and the corresponding tag information, of the cache line are provided to the selection logic 30. In the event the instruction cache 16 includes more than one set, then the selection logic 30 selects the proper line from the plurality of sets. With embodiments having only a single set, the selection logic 30 simply passes the accessed line 34 to the instruction prefetch and dispatch unit 12. The set selection logic 30 is intended to represent a broad category of selection logic found in most computer systems, including the selection logic described in U.S. patent application, Ser. No. 07/906,699, filed on Jun. 30, 1992, now U.S. Pat. No. 5,392,414, entitled Rapid Data Retrieves From A Data Storage Using Prior Access Predictive Annotation assigned to the same assignee of the present invention.

[0027] The BRPD fields 40 and NFAPD fields 42 are initialized in accordance with a pre-established policy when a cache line 34 is brought into the instruction cache 16. When an instruction is fetched, the corresponding ICLASS field 44 information, BRPD field 40 information and the NFAPD field 42 information are all provided to the prefetch and dispatch unit 12. Since the instruction has already been decoded to determine class, the need to perform a full decode in the prefetch and dispatch unit 12 to determine if an instruction is a branch instruction is avoided. If the instruction is a non-branch instruction, the BRPD information is ignored. The NFAPD information, however, provides the next address to be fetched, which is typically the sequential address of the next line in the instruction cache 16. If a predecoded instruction is a branch instruction, the corresponding BRPD field 40 contains either a "Branch
Taken” or a “Branch Not Taken” prediction and the NFAPD field 42 contains a prediction of either the target address of the branch instruction or the sequential address of the next line 34 in the instruction cache 16. Regardless of the type of instruction, the predicted next address is used to immediately fetch the next instruction.

[0028] After a branch instruction is fetched, an update policy is used to update the entries in the corresponding BRPD field 40 and the NFAPD field 42 when the actual direction of the branch instruction and the actual next fetch address is resolved in the execution units 14. If the branch prediction and next fetch address prediction were correct, execution continues and the BRPD field 40 or the NFAPD field 42 are not altered. On the other hand, if either prediction is wrong, the BRPD field 40 and the NFAPD field 42 are updated as needed by the prefetch and dispatch unit 12. If the misprediction caused the execution of instructions down an incorrect branch path, execution is stopped and the appropriate execution units 14 are flushed. Execution of instructions thereafter reselections along the correct path. The next time the same instruction is fetched, a branch prediction decision is made based on the updated branch prediction information in the BRPD field 40 and the next prefetch address is based on the updated contents of NFAPD field 42.

[0029] During operation, the BRPD fields 40 and NFAPD fields 42 are updated in accordance with a specified update policy. For the sake of simplicity, only a single bit of information is used for the BRPD field 40. This means that the BRPD field 40 can assume one of two states, either “Branch Taken” or “Branch Not Taken”. One possible update policy is best described using a number of examples, as provided below.

[0030] 1. If the BRPD predicts “Branch Taken” and the NFAPD field contains the target address, and the actual branch is not taken, then the BRPD is updated to “Branch Not Taken” and the NFAPD is updated to the next sequential address.

[0031] 2. If the BRPD predicts “Branch Taken”, and the actual branch is taken, but the NFAPD misses, then the NFAPD is updated to the target address of the branch instruction.

[0032] 3. If the BRPD predicts “Branch Not Taken” and the NFAPD field contains the next sequential address, and the actual branch is taken, then the BRPD is updated to “Branch Taken” and the NFAPD is updated to the target address of the branch instruction.

[0033] 4. If the BRPD predicts “Branch Not Taken”, and the actual branch is not taken, but the NFAPD misses, the NFAPD is updated to the sequential address.

[0034] 5. If the BRPD predicts “Branch Not Taken”, and the actual branch is not taken, and the NFAPD provides the next sequential address, then the BRPD and NFAPD fields are not updated.

[0035] 6. If the BRPD predicts “Branch Taken” and the actual branch is taken and the NFAPD provides the target address, then the BRPD and NFAPD fields are not updated.

[0036] In summary, the BRPD field and the NFAPD field are updated to the actual branch taken and actual next fetch address. In alternative embodiments, more sophisticated branch prediction algorithms may be used. For example, multiple bits may be used for the BRPD field 42, thereby providing finer granularity and more information about each branch prediction.

[0037] In one embodiment, a specific BRPD field 40 and a corresponding NFAPD field 42 is provided for each instruction per cache line 34 (i.e., n=m=k). As such, each branch instruction per cache line 34 enjoys the benefit of a dedicated branch prediction and next fetch address prediction as stored in BRPD field 40 and corresponding NFAPD field 42 respectively. In a simplified embodiment, one BRPD field 40 (i.e., n=1) and one NFAPD field 42 (i.e., k=1) is shared among all the instructions per cache line 34. With this embodiment, only the dominant instruction in the cache line 34 makes use of the branch prediction information and the next fetch address information. A dominant instruction is defined as the first branch instruction with a “Branch Taken” status in the cache line 34. Therefore, the BRPD contains a “Branch Taken” prediction and the corresponding NFAPD typically contains the target address for the dominating instruction. Since the dominant instruction is the first instruction in the cache line to cause a control transfer, it is not necessary for the other instructions to have their own BRPD fields 40 and NFAPD fields 42.

[0038] It will be appreciated that the number of BRPD fields 40 and NFAPD fields 42 is design dependent. As the number of BRPD fields 40 (m) and NFAPD fields 42 (k) increases toward the number of instructions (n) per cache line 34, the likelihood of branch and next fetch address prediction hits will increase. In contrast, as the number of BRPD fields 40 and NFAPD fields 42 approaches one, the likelihood of mispredictions increases, but the structure of cache 16 is simplified.

[0039] Referring to FIG. 3, a block diagram of the pertinent sections of the prefetch and dispatch unit 12 are shown. The prefetch and dispatch unit 12 includes a comparator 68, a next fetch address (NFA) register 70, an instruction queue 72, an update unit 74, and a dispatch unit 76. For each instruction, the comparator 68 is coupled to receive the BRPD field 40 and the NFAPD field 42 information from instruction cache 16 and the actual branch direction and next fetch address from the execution units 14. It should be noted that the actual branch and next fetch address typically arrive at the comparator 68 at a later point in time since a certain period of time is needed for the actual branch to resolve in the execution units 14. The comparator 68 determines if the BRPD and the NFAPD are respectively correct, i.e., a hit. If the comparison yields a miss, the BRPD field and/or the NFAPD field 42 information is updated by update circuit 74 in accordance with the update policy described above. The updated BRPD and/or NFAPD information is then returned to the instruction cache 16. The actual NFA also is placed in the NFA register 70.

[0040] Referring now to FIG. 4a and FIG. 4b, two flow diagrams illustrating the operation of the prefetch and dispatch until 12 are shown. In FIG. 4a, the instruction prefetch and dispatch unit 12 determines if a fetch/prefetch should be initiated (block 94), if a fetch/prefetch should be initiated, the instruction prefetch and dispatch unit 12 uses the address stored in the NFA register 70 to fetch the next instruction from instruction cache 16 (block 96). In response, the instruction cache 16 provides the instruction prefetch and dispatch unit 12 with the requested instruction. The instruc-
tion is then placed into the instruction queue 72. Thereafter, the instruction is dispatched by dispatch unit 76. It should be noted that with each fetched instruction, the corresponding NFAPD value is placed in the NFA register 70 and is used to fetch the next instruction. When the comparator 68 determines that the NFAPD is incorrect, the actual NFA is placed into the NFA register 70, and the fetching of instructions resumes at the actual NFA. The instruction prefetch and dispatch unit repeats the above process steps until the instruction queue 72 is empty or the computer system is shut down.

[0041] As shown in FIG. 4, the instruction prefetch and dispatch unit 12 also receives a branch resolution signal 200 (actual branch) as the branch instruction completes execution in the execution units 14 (block 108). The instruction prefetch and dispatch unit 12 then determines if the branch prediction is correct (diamond 110). If the predicted branch is incorrect, the instruction prefetch and dispatch unit 12 updates the selected BRPD field 40 and the NFAPD field 42 in accordance with the above-defined update policy (block 114). If the selected BRPD predicted the branch direction correctly, the instruction prefetch and dispatch unit 12 determines if the next address in the NFAPD field is correct (block 112). If the selected NFAPD predicted the next fetch address incorrectly, the instruction prefetch and dispatch unit 12 updates the NFAPD (block 116). If the NFAPD is correct, its status remains unchanged.

[0042] Referring now to FIG. 5, a flow diagram illustrating the operation of the instruction cache 16 is shown. The instruction cache 16 receives the fetch address from the instruction prefetch and dispatch unit 12 (block 74). In response, the instruction cache 16 determines if there is a cache hit (block 76). If there is a cache hit, select logic 30, if necessary, selects and provides the appropriate set of instructions and the corresponding ICLASS field 44, BRPD field 40 and NFAPD field 42 information to the instruction prefetch and dispatch unit 12.

[0043] If there is a cache miss, the instruction cache 16 initiates a cache fill procedure (block 80). In one embodiment, the instructions accessed from memory 20 are provided directly to prefetch and dispatch unit 12. Alternatively, the instructions may be provided to the instruction prefetch and dispatch unit 12 after the cache line is filled in cache 16. As described earlier, the instructions are decoded to determine their class prior to being stored in the instruction cache 16. Additionally, the BRPD field 40 and NFAPD field 42 are initialized in accordance with the initialization policy of the branch and next fetch address prediction algorithm (block 86).

Operation

[0044] For the purpose of describing the operation of the present invention, several examples are provided. In the provided examples, there is only one (1) BRPD field 40 and NFAPD field 42 provided per cache line (i.e., m=k=1). For the purpose of simplifying the examples, the BRPD field 42 contains only 1 bit of information, and therefore can assume only two states: “Branch Taken” and “Branch Not Taken”.

[0045] Referring to FIG. 6, several lines 34, 34, of the instruction cache 16 is shown. In this example, there are four instructions (n=4) per cache line 34. The four instructions are labeled, from left to right 4, 3, 2, 1, respectively, as illustrated in column 101 of the cache 16. A “1” bit indicates that the instruction in that position is a branch instruction. A “0” bit indicates that the instruction is some other type of instruction, but not a branch instruction. In column 103, the BRPD fields 40 for the cache lines 34 are provided. A single BRPD field 40 (m=1) is provided for the four instructions per cache line 34. In the BRPD field 40, a “0” value indicates a “Branch Not Taken” prediction and a “1” value indicates “Branch Taken” prediction. With this embodiment, the BRPD information provides the branch prediction only for the dominant instruction in the cache line. The column 105 contains the next fetch address in the NFAPD field 42. A single NFAPD field 42 (k=1) is provided for the four instructions per cache line 34. If the BRPD field 40 is set to “0”, then the corresponding NFAPD field 42 contains the address of the next sequential instruction. On the other hand, if the BRPD field 40 contains a “1”, then the corresponding NFAPD field 42 contains the target address of the dominant instruction in the cache line 34.

[0046] In the first cache line 34, the four instructions are all non-branch instructions, as indicated by the four “0” in column 101. As such, the corresponding BRPD field 40 is set to “0” “Branch Not Taken” and the NFAPD field 42 is set to the sequential address.

[0047] The second and third cache lines 34, and 34, each include one branch instruction respectively. In the cache line 34, the branch instruction is located in the first position, as indicated by the “1” in the first position of column 101. The corresponding BRPD field is set to “0”, and NFAPD is set to “next addr +1”. Accordingly, the branch prediction is “Branch Not Taken”, and the NFAPD is the next sequential address (i.e., 34). In the third cache line 34, the first instruction is a branch instruction. The corresponding BRPD field is set to “1”, and NFAPD is set to “target addr +1”. The branch prediction algorithm thus predicts “Branch Taken”, and the next fetch address is the “target address +1” of the first instruction.

[0048] The fourth cache line 34, and fifth cache line 35, provide examples of cache lines 34 having two branch instructions. In both lines 34, and 34, the branch instructions are located in the first and third positions in column 101. With cache line 34, both instructions have a branch prediction set to “Branch Not Taken”, i.e., there are no dominant instructions. The corresponding field BRPD is therefore set to “0”, and NFAPD is set to “next addr +2”.

[0049] In contrast, with the fifth cache line 35, the branch prediction algorithm predicts “Branch Taken” for the first branch instruction. The first instruction in the cache 35, is therefore the dominant instruction of the cache line. The corresponding BRPD field is set to “1”, and NFAPD is set to “target addr +1”. Since the dominant instruction will cause a control transfer, the branch prediction and next fetch address for the third instruction are not necessary. The sixth 34, and seventh 34, cache lines provide two more examples of cache lines having two branch instructions. In both cache lines, the first and third instruction are branch instructions. In the sixth cache line 34, the branch prediction is “Branch Not Taken”, but the prediction for the second branch instruction is “Branch Taken”. Accordingly, the third instruction is considered the dominant instruction and the NFAPD field contains the target address for the third instruction of the line. Thus, BRPD is set to “1”, and NFAPD is set to “target
address 3°. In the seventh cache line 34, the branch prediction for both branch instructions is "Branch Taken". Since the first instruction is the dominant instruction of the line, the BRPD field is set to "Branch Taken" and the NFPAD field is set to "target addr 1".

[0050] In embodiments where the number of BRPD fields 40 and NFPAD fields 42 equals the number of instructions per cache line 34 (i.e., m+n), the operation of the present invention is straightforward. The BRPD field 40 and the NFPAD field 42 for each branch instruction are used to predict the "Branch Taken" and next fetch address. Further, the BRPD field 40 and the NFPAD field 42 are updated in accordance with the outcome of the respective branch instruction when executed.

[0051] While the invention has been described in relationship to the embodiments shown in the accompanying figures, other alternatives, embodiments and modifications will be apparent to those skilled in the art. It is intended that the specification be only exemplary, and that the true scope and spirit of the invention be indicated by the following claims.

What is claimed is

1. In a computer system comprising at least one execution unit for executing instructions, a method for rapidly dispatching instructions to said at least one execution unit for execution, said method comprising the steps of:
   a) storing a plurality of sets of instructions in a plurality of cache lines of an instruction cache array;
   b) storing a plurality of corresponding sets of tag and associated control information in a plurality of corresponding tag entries of a corresponding tag array;
   c) storing a plurality of corresponding sets of instruction classes in a plurality of corresponding instruction class entries of a corresponding instruction class array, each of said set of instruction classes comprising a plurality of instruction classes for said instructions of said corresponding set of instructions;
   d) storing a plurality of corresponding sets of predictive annotations in a plurality of corresponding predictive annotation entries of a corresponding predictive annotation array, each of said set of predictive annotations comprising at least one branch prediction for said instructions of said corresponding set of instructions; and
   e) fetching and prefetching repeatedly selected ones of said stored sets of instructions for dispatch to said at least one execution unit for execution using said stored corresponding instruction classes and branch predictions.

2. The method as set forth in claim 1, wherein, said instruction class and predictive annotation entries are stored into said corresponding instruction class and predictive annotation arrays in said steps c) and d) one instruction class and corresponding predictive annotation entry at a time, each of said instruction class and corresponding predictive annotation entries being stored into said instruction class and predictive annotation arrays when their corresponding cache line of instructions is stored into said instruction cache array, each of said branch predictions of said predictive annotation entries being initialized in accordance with an initialization policy of a branch prediction algorithm when its predictive annotation entry is stored into said predictive annotation array.

3. The method as set forth in claim 2, wherein, each of said at least one branch prediction of each of said sets of predictive annotations is initialized to predict "branch will not be taken".

4. The method as set forth in claim 1, wherein, each of said fetchings and prefetchings in said step e) comprises the steps of:
   a)1) accessing one of said cache line of instructions and its corresponding tag, instruction class and predictive annotation entries concurrently using a fetch address;
   b)2) selecting one of said sets of instructions from said accessed cache line and a branch prediction from said selected set of instructions' corresponding set of predictive annotations in said accessed cache line's corresponding predictive annotation entry;
   c)3) determining a next fetch address from said selected branch prediction;
   d)4) determining subsequently whether said selected branch prediction predicts correctly; and
   e)5) updating said selected branch prediction in accordance to an update policy of a branch prediction algorithm based on said prediction correctness determination.

5. The method as set forth in claim 4, wherein, said update policy in said step e) is:
   e)5) updates each of said selected branch predictions as follows:

<table>
<thead>
<tr>
<th>Branch Class</th>
<th>Prediction, Actual</th>
<th>Update Policy</th>
</tr>
</thead>
<tbody>
<tr>
<td>PC-relative branch</td>
<td>PT, ANT</td>
<td>PNT -&gt; BRPD[A]</td>
</tr>
<tr>
<td>PC-relative branch</td>
<td>PT, AT</td>
<td>PT -&gt; BRPD[A]</td>
</tr>
<tr>
<td>PC-relative branch</td>
<td>PT, AT</td>
<td>No Action</td>
</tr>
<tr>
<td>PC-relative branch</td>
<td>PT, AT</td>
<td>PT -&gt; BRPD[A]</td>
</tr>
<tr>
<td>Register indirect</td>
<td>PT, AT</td>
<td>No Action</td>
</tr>
<tr>
<td>Unconditional PC</td>
<td>PT, AT</td>
<td>PT -&gt; BRPD[A]</td>
</tr>
<tr>
<td>Unconditional PC</td>
<td>PT, AT</td>
<td>No Action</td>
</tr>
</tbody>
</table>

6. The method as set forth in claim 1, wherein, each of said set of predictive annotations in said step d) further comprises at least one next fetch address prediction for said instructions of said corresponding set of instructions; and

7. The method as set forth in claim 6, wherein, said next fetch address predictions are initialized, accessed, selected, and updated in substantially the same manner as said branch predictions.
8. The method as set forth in claim 7, wherein, each of said at least one next fetch address prediction of each of said sets of predictive annotations is initialized to predict an address that equals to a sum of a program counter and a next sequential fetch block size, said program counter indicating a current fetch address and said next fetch sequential block size indicating a next sequential fetch block’s block size.

9. The method as set forth in claim 7, wherein, each of said selected next fetch address predictions is updated as follows:

<table>
<thead>
<tr>
<th>Branch Type</th>
<th>Prediction, Actual</th>
<th>Next Fetch Addr Hit?</th>
<th>Update Policy</th>
</tr>
</thead>
<tbody>
<tr>
<td>PC relative branch</td>
<td>PT, ANT</td>
<td>A + FS = NFAPD[A]</td>
<td></td>
</tr>
<tr>
<td>PC relative branch</td>
<td>PT, AT</td>
<td>TA → NFAPD[A]</td>
<td></td>
</tr>
<tr>
<td>PC relative branch</td>
<td>PNT, AT</td>
<td>TA → NFAPD[A]</td>
<td></td>
</tr>
<tr>
<td>PC relative branch</td>
<td>PNT, ANT</td>
<td>A + FS = NFAPD[A]</td>
<td></td>
</tr>
<tr>
<td>PC relative branch</td>
<td>PT, AT</td>
<td>Hit</td>
<td>No Action</td>
</tr>
<tr>
<td>PC relative branch</td>
<td>PNT, ANT</td>
<td>Hit</td>
<td>No Action</td>
</tr>
<tr>
<td>Register indirect control transfer</td>
<td>PT, AT</td>
<td>TA → NFAPD[A]</td>
<td></td>
</tr>
<tr>
<td>Register indirect control transfer</td>
<td>PT, AT</td>
<td>TA → NFAPD[A]</td>
<td></td>
</tr>
<tr>
<td>Register indirect control transfer</td>
<td>PT, AT</td>
<td>Hit</td>
<td>No Action</td>
</tr>
<tr>
<td>Unconditional PC control transfer</td>
<td>PNT, AT</td>
<td>TA → NFAPD[A]</td>
<td></td>
</tr>
<tr>
<td>Unconditional PC control transfer</td>
<td>PT, AT</td>
<td>TA → NFAPD[A]</td>
<td></td>
</tr>
<tr>
<td>Unconditional PC control transfer</td>
<td>PT, AT</td>
<td>No Action</td>
<td></td>
</tr>
</tbody>
</table>

d) predictive annotation array means comprising a plurality of predictive annotation entries for storing a plurality of corresponding sets of predictive annotations, each of said set of predictive annotations comprising at least one branch prediction for said instructions of said corresponding set of instructions; and
e) fetching and prefetching means coupled to said instruction array means, said tag array means, said instruction class array means, and said predictive annotation array means for fetching and prefetching repeatedly selected ones of said stored sets of instructions for dispatch to said at least one execution unit for execution using said stored corresponding instruction classes and branch predictions.

10. The method as set forth in claim 7, wherein,
each set of said predictive annotations comprises one branch prediction and one next fetch address prediction, said one branch and next fetch address predictions predicting branch direction and next fetch address for a dominant instruction of its corresponding set of instructions;
said method further comprises the step of f) storing each of said selected next fetch address predictions in a register, said register being used for storing a next fetch address for a next set of instructions to be fetched, said stored next fetch address being also used for selecting said branch prediction and said next fetch address prediction for said next set of instructions to be fetched.

11. In a computer system comprising at least one execution unit for executing instructions, an apparatus for rapidly dispatching instructions to said at least one execution unit for execution, said apparatus comprising:
a) instruction array means comprising a plurality of cache lines for storing a plurality of sets of instructions;
b) tag array means comprising a plurality of tag entries for storing a plurality of corresponding sets of tag and associated control information;
c) instruction class array means comprising a plurality of instruction class entries for storing a plurality of corresponding sets of instruction classes, each of said set of instruction classes comprising a plurality of instruction classes for said instructions of said corresponding set of instructions;
d) predictive annotation array means comprising a plurality of predictive annotation entries for storing a plurality of corresponding sets of predictive annotations, each of said set of predictive annotations comprising at least one branch prediction for said instructions of said corresponding set of instructions; and
e) fetching and prefetching means coupled to said instruction array means, said tag array means, said instruction class array means, and said predictive annotation array means for fetching and prefetching repeatedly selected ones of said stored sets of instructions for dispatch to said at least one execution unit for execution using said stored corresponding instruction classes and branch predictions.

12. The apparatus as set forth in claim 11, wherein,
said instruction class and said predictive annotation array means store each of said instruction class and corresponding predictive annotation entries one instruction class and corresponding predictive annotation entry at a time,
said instruction class and predictive annotation array means stores each of said instruction class and corresponding predictive annotation entries into said instruction class and predictive annotation array means when said instruction array means stores its corresponding cache line of instructions into itself,
said predictive annotation array means initializes each of said branch predictions of said predictive annotation entries in accordance to an initialization policy of a branch prediction algorithm when said predictive annotation array means stores its predictive annotation entries into itself.

13. The apparatus as set forth in claim 12, wherein, said predictive annotation array means initializes each of said at least one branch prediction of each of said sets of predictive annotations to predict “branch will not be taken”.

14. The apparatus as set forth in claim 11, wherein, said fetching and prefetching means comprises:
c.1) accessing means for accessing one of said cache line of instructions and its corresponding tag, instruction class and predictive annotation entries stored in said instruction, tag, instruction class and predictive annotation array means concurrently using a fetch address;
c.2) selection means coupled to said instruction, tag, instruction class and predictive annotation array means for selecting one of said sets of instructions from said accessed cache line and a branch prediction from said selected set of instructions’ corresponding set of predictive annotations in said accessed cache line’s corresponding predictive annotation entry;
c.3) first determination means coupled to said selection means for determining a next fetch address from said selected branch prediction;
c.4) second determination means coupled to said selection means and said execution means for determining subsequently whether said selected branch prediction predicts correctly; and
c.5) update means coupled to said second determination means and said instruction, tag and predictive annotation array means for updating said selected branch...
prediction in accordance with an update policy of a branch prediction algorithm based on said prediction correctness determination.

15. The apparatus as set forth in claim 14, wherein, said update means updates each of said selected branch predictions as follows:

<table>
<thead>
<tr>
<th>Branch Type</th>
<th>Prediction, Actual</th>
<th>Update Type</th>
<th>Update Policy</th>
</tr>
</thead>
<tbody>
<tr>
<td>PC relative branch PT, ANT</td>
<td>PNT -&gt; BRPD[A]</td>
<td>PT, AT</td>
<td>A + FS = NFAPD[A]</td>
</tr>
<tr>
<td>PC relative branch PNT, AT</td>
<td>PT -&gt; BRPD[A]</td>
<td>No Action</td>
<td>TA -&gt; NFAPD[A]</td>
</tr>
<tr>
<td>PC relative branch PNT, AT</td>
<td>No Action</td>
<td>TA -&gt; NFAPD[A]</td>
<td></td>
</tr>
<tr>
<td>Register Indirect control transfer PNT, AT</td>
<td>PT, AT</td>
<td>No Action</td>
<td>TA -&gt; NFAPD[A]</td>
</tr>
<tr>
<td>Unconditional PC control transfer PNT, AT</td>
<td>PT -&gt; BRPD[A]</td>
<td>No Action</td>
<td>TA -&gt; NFAPD[A]</td>
</tr>
</tbody>
</table>

16. The apparatus as set forth in claim 11, wherein, each of said set of predictive annotations further comprises at least one next fetch address prediction for said instructions of said corresponding set of instructions; and said fetching and prefetching means uses said stored corresponding next fetch address predictions as well as said instruction classes and branch predictions.

17. The apparatus as set forth in claim 16, wherein, said fetching and prefetching means comprises accessing mean, selection means and update means for accessing, selecting and updating said branch predictions, said predictive annotation array means, said accessing means, said selection means, and said update means initializes, accesses, selects, and updates said next fetch address predictions in substantially the same manner as said branch predictions.

18. The apparatus as set forth in claim 17, wherein, said predictive annotation array means initializes each of said at least one next fetch address prediction of each of said sets of predictive annotations to predict an address that equals to a sum of a program counter and a next sequential fetch block size, said program counter indicating a current fetch address and said next sequential fetch block size indicating a next sequential fetch block's block size.

19. The apparatus as set forth in claim 17, wherein, said update means update each of said selected next fetch address predictions as follows:

<table>
<thead>
<tr>
<th>Branch Type</th>
<th>Prediction, Actual</th>
<th>Next Fetch Addr Hit/ Miss</th>
<th>Update Policy</th>
</tr>
</thead>
<tbody>
<tr>
<td>PC relative branch PNT, AT</td>
<td>PT -&gt; BRPD[A]</td>
<td>TA -&gt; NFAPD[A]</td>
<td>No Action</td>
</tr>
<tr>
<td>PC relative branch PNT, AT</td>
<td>No Action</td>
<td>TA -&gt; NFAPD[A]</td>
<td>No Action</td>
</tr>
<tr>
<td>Register Indirect control transfer PNT, AT</td>
<td>PT, AT</td>
<td>No Action</td>
<td>TA -&gt; NFAPD[A]</td>
</tr>
<tr>
<td>Unconditional PC control transfer PNT, AT</td>
<td>PT -&gt; BRPD[A]</td>
<td>No Action</td>
<td>TA -&gt; NFAPD[A]</td>
</tr>
</tbody>
</table>

20. The apparatus as set forth in claim 17, wherein, each set of said predictive annotations comprises one branch prediction and one next fetch address prediction, said one branch and next fetch address predictions predicting branch direction and next fetch address for a dominant instruction of its corresponding set of instructions; said apparatus further comprises e) register means coupled to said fetching and prefetching means for storing each of said selected next fetch address predictions, said register being used for storing a next fetch address for a next set of instructions to be fetched, said stored next fetch address being also used for selecting said branch prediction and said next fetch address prediction for said next set of instructions to be fetched.

* * * * *