Description

This invention relates to the operation of an instruction prefetch buffer in a pipelined processor, and more particularly to a method and an apparatus for prefetching and executing a plurality of instructions having at least one branch instruction.

In a pipelined processor, multiple pieces of data, such as instructions, can move simultaneously through the channel connecting the processor with a storage means. A storage controller is used to direct such data between the storage means and the processor through the storage channel. The data moving through the channel is tagged so as to identify its destination in the processor. Since the operation of the processor is much faster than that of the storage means, the use of such a pipelined storage channel together with appropriate buffers within the processor permits the processor to operate at its most effective rate and to conduct a plurality of simultaneous storage transactions. The concept of a pipelined storage channel is described in IBM Technical Disclosure Bulletin article, "Synchronous LSSD Packet Switching Memory and I/O Channel", published March 1982, pages 4986-4987 by Jeremiah et al, and in IBM Technical Disclosure Bulletin article, "Exact Interrupt Capability for Processors Using a Packet-Switching Storage Channel", published August 1982, pages 1771-1772 by Hester et al.

Instructions in such a pipelined processor are usually fetched several cycles in advance of their execution. This increases processor performance by allowing instruction accessing to be overlapped with instruction execution. A problem arises, however, whenever a successful branch instruction is encountered. A successful branch invalidates all instructions which have been prefetched since they are part of an old instruction stream. During the time that the processor is waiting to access the new instruction stream for the branch, it is essentially idle. This idle time decreases the overall performance of the processor.

One technique for overcoming the branching problem is disclosed in U. S. Patent 4,430,706, wherein a branch prediction approach is employed. In this approach, an auxiliary implementation is added to a pipelined data processor for monitoring both the instruction flow and the recent conditional branches and their outcome. Whenever a branch instruction is encountered more than once, its prior behavior is used to predict the particular branch to be taken. The processor then provisionally executes the instructions in such predicted branch. However, when a predicted branch turns out to be incorrect, such processing ceases and the processor then attempts to take a correct branch. Two problems with this approach are that, first, extra hardware is required, and second, processing time is wasted whenever a wrong branch is predicted and subsequently processed.

Another approach to the branching problem is disclosed in U. S. Patents 4,155,120 and 4,179,737, wherein it is assumed that no branching possibilities are present. In accordance with this approach, microinstruction sequencing is assumed to be unconditional even though a plurality of branching possibilities may indeed exist in the microinstruction sequencing flow. In this manner, microinstruction sequencing may proceed rapidly as long as no branching occurs. However, when a branch does in fact occur, it must be detected and then a correction cycle must be initiated. A problem with this approach is that as the number of branching operations increases, sequencing must be altered and corrected for incorrect sequencing because of lack of recognition of branching possibilities. Further this approach requires means for monitoring the correctness of the sequencing of microinstructions concurrently with the execution of microinstructions during each cycle for which a branching decision is required.

A further problem that arises as a result of a branch is that the instruction prefetch buffer in the processor has no advance knowledge of a successful branch, so there may be several pending requests from previous prefetches when the successful branch occurs. The problem then is how to distinguish between old prefetches, i.e., ones that occurred before the successful branch and which are no longer necessary, and the new prefetches which are needed as a result of the successful branch.

One approach to solve this branching problem is to divide the instruction prefetch buffer into two halves, and to alternate between the two halves every time a successful branch occurs. Since the storage channel utilizes a tag to identify each request source, a set of tags could be associated with each half of the instruction prefetch buffer. Thus, whenever a successful branch occurs, the instruction prefetch buffer would change to the alternate set of tags. In this manner, the instruction prefetch buffer could determine whether or not a reply from a prefetch request should go into the active half of the buffer by comparing the returned tag with the currently active tag set. If the tag were a member of the active set, then the prefetch would be associated with the current execution stream and consequently should be placed in the active half of the buffer. Conversely, if the tag were not a member of the active set, then it would no longer be required and could be placed in the inactive half of the buffer. Although this approach is workable, it only utilizes one-half of the storage area in the instruction prefetch buffer at any given

time.

Accordingly, it is an object of this invention to provide improved execution sequencing in a pipelined processor system.

It is another object of this invention to provide improved instruction prefetch buffer utilization and efficiency.

Accordingly, a method and an apparatus are set out in claims 1 and 6, respectively.

This technique reduces the amount of storage required in the instruction prefetch buffer (IPB) by a factor of two without reducing IPB performance. The technique is especially useful for handling branching operations. When a branch occurs, the IPB begins fetching instructions starting at the branch target address. However, since the IPB has no advance knowledge of the branch, there may be several pending requests from the previous prefetched instructions when the branch occurs. The technique disclosed herein enables the IPB to distinguish between old prefetches, that is, those that occurred before the branch, and the new prefetches which are needed as a result of the branch.

The IPB contains a plurality of registers which are used to hold instructions prior to their execution. Each of these registers has a number associated with it which uniquely identifies that register. Additionally, a control bit is associated with each register to indicate if there is a cancelled fetch outstanding (CFO) for that particular register. A cancelled fetch outstanding occurs after a branch when a previous fetch has not yet been returned to the IPB from storage over the storage channel. Recall that these previous prefetches are no longer required after the branch occurs. In addition to the register identifier and the CFO control bit, there is also a control bit in the processor which is complemented every time a branch occurs. This latter control bit is known as the branch target bit (BTB).

When a prefetch is returned to the IPB, the BTB value associated with the returning prefetch is compared to the current BTB value in the IPB. If the two are equal, then this is an indication that a branch has not occurred and that the prefetch request which was sent to storage is still needed. In this situation, the returning prefetch is written into the register in the IPB specified by control bits of the returning prefetch. The returned instruction is then used when needed for execution. However, if the BTB value associated with the returning prefetch is not equal to the current BTB value in the IPB, then this is an indication that a branch has occurred and that the returning prefetch is no longer needed. As a result, the returning prefetch is not written into one of the registers of the IPB.

Whenever a branch occurs, the CFO bits associated with each register of the IPB are examined. If any of such bits are set, then the branch must wait until all CFO bits are reset. As previously noted, the BTB is also complemented when the branch occurs. Finally, CFO bits in the IPB are set for locations for which a prefetch has been sent to storage but has not yet returned.

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

Fig. 2 is a flow chart depicting the branching operation according to the present invention.

Fig. 3 is a flow chart depicting instruction execution according to the present invention.

Fig. 4 is a flow chart depicting the prefetching operation of the present invention.

Fig. 5 is a flow chart depicting the returning tag operation of the present invention.

Referring now to Fig. 1, there is shown a pipelined processor with particular emphasis on instruction prefetch buffer 11. Information in the form of instructions is communicated over storage channel 12 through storage controller 13 between instruction prefetch buffer (IPB) 11 and storage 14. IPB 11 comprises four registers, IPB0, 15, IPB1, 16, IPB2, 17, and IPB3, 18. Each of registers 15 through 18 has logic means, CFO, associated with it for indicating whether each of these registers has a cancelled fetch outstanding (CFO). IPB 11 also includes a branch tag register (BTR) 19 which contains a control bit for indicating whenever an instruction branch occurs. IPB 11 also contains control logic 21 for directing the fetching of instructions to and from IPB 11.

Each of registers 15 through 18 has a number associated with it, 0 through 3, which uniquely identifies such register. Additionally, there is a control bit associated with each of registers 15 through 18 to indicate if there is a cancelled fetch outstanding (CFO) for that associated register. There is also a control bit in BTR 19 called the branch target bit (BTB) which is complemented every time a branch occurs in the instruction stream.

In operation, whenever the next instruction is to be fetched, IPB 11 determines which of registers 15 through 18 should be used for the fetch. A tag is then generated for the prefetch so as to indicate the source of the instruction request. Control logic 21 generates this tag for the instruction fetch by appending the current BTB value to the register number selected for such request. The prefetch request is then sent to storage 14 over storage channel 12 in order to obtain the next instruction.

After the prefetch request has been processed by storage 14, it will be returned over channel 12 to IPB 11. The returned prefetch request will contain the next instruction along with its identifying tag. A determination will then be made if the returned prefetch instruction is still needed. This is done by comparing the BTB value in the returning

instruction to the current BTB value in BTR 19. If the two values are equal, then this is an indication that a branch has not occurred since the time that the prefetch request was sent to storage 14. Consequently, the prefetch instruction is still required. At this time, the prefetch instruction returned is then written into the register specified by the returning tag identifier, and the instruction is then used when it is needed for an execution.

If the BTB value in the returning prefetch instruction and the current BTB value in BTR 19 are different, then this is an indication that a branch in the instruction stream has occurred. Consequently, the prefetch instruction is no longer needed and therefore, is not written in one of registers 15 through 18. The CFO bit associated with the returning prefetch instruction is then reset. At the time a branch in the instruction stream occurs, the CFO bits associated with each of registers 15 through 18 are examined. If any of these CFO bits are set, then the execution of the branch in the instruction stream must wait until all of these CFO bits are reset. Also at this time, the BTB in BTR 19 is complemented. Finally, control logic 21 examines each of registers 15 through 18 to determine if any prefetch has been sent to storage 14 but has not yet returned. If this is the case, then the CFO bit associated with that register is set thus indicating that there is a cancelled fetch outstanding.

The flow chart in Fig. 2 details the operation of IPB 11 at the time a branch in the instruction stream occurs. As indicated in box 22, a determination is made as to whether or not there are any cancelled fetches outstanding. If there are, then box 23 indicates that no further operation occurs until the cancelled tag identifying a returned prefetch instruction is sent from storage 14 to IPB 11. If there are no current fetches outstanding, then box 24 provides that control logic 21 sets a CFO bit for all registers 15 through 18 for which a prefetch has been allocated. Also at this time, the BTB value in BTR 19 is complemented as indicated in box 25 and all tags are set to their available state for enabling instructions to be fetched from storage as indicated in box 26. Finally in box 27, an identifying tag is selected in order to fetch the branch target instruction from storage 14. This prefetching process continues until all target fetches are completed.

The flow chart in Fig. 3 indicates the operation of IPB 11 when instruction execution unit 37 asks for the next instruction. As indicated in box 31, the next instruction address is calculated within IPB 11. Then in box 32, a decision is made as to whether or not the location in IPB 11 of the next instruction has been freed. If the answer is affirmative, then the freed IPB tag is set to its available state. However, if the answer is negative, then a further decision is made in block 34 as to whether or not the next instruction tag is in a valid state. If the next instruction tag is not in a valid state, then IPB 11 must wait for the selected tag to return from storage 14 over channel 12. However, if the next instruction tag is in a valid state, then the next instruction is provided to execution unit 37 for execution.

The process for selecting the next identifying tag for a prefetch is detailed in the flow chart shown in Fig. 4. After the next prefetch tag has been selected, a determination is made as to whether or not the tag is available. If such tag is not available, then IPB 11 must wait for such tag to become available as indicated in blocks 42 and 43. However, if the tag is available, then it is allocated to the next prefetch as shown in block 44. Finally the next prefetch along with identifying tag is sent to storage 14 so as to obtain the next instruction.

As shown in Fig. 5, when an instruction is returned from storage 14 with its associated identifying tag, a determination is made as to whether or not the BTB returning with the instruction is equal to the current BTB value in IPB 11. If the two values are not equal, then this is an indication that a branch has occurred in the instruction stream and consequently the prefetched instruction is no longer needed. The CFO bit for each returning tag is then reset. However, if the returning BTB value is equal to the current BTB value in IPB 11, then as stated previously this is an indication that a branch in the instruction stream has not occurred. A determination is then made as shown in block 53, as to whether or not a reply to an exception state exists. If the answer is affirmative then the tag identifier is set to the exception state, while if the answer is negative then the tag is set to the valid state and the instruction awaits execution.

Claims

1. A method for prefetching and executing a plurality of instructions having at least one branch instruction in a pipelined system comprising an instruction prefetch buffer (11) having a plurality of prefetch registers (15-18), a control logic (21) for directing the fetching of instructions to and from said prefetch registers, a storage (14), and a storage channel (12) for the exchange of instructions between said registers and said storage ;

said method being characterized in that it comprises the steps of :

generating a control tag (BTB) from said control logic each of whose current value is changed when a successful branch is execut-

* C L M S *

instruction to the current BTB value in BTR 19. If the two values are equal, then this is an indication that a branch has not occurred since the time that the prefetch request was sent to storage 14. Consequently, the prefetch instruction is still required. At this time, the prefetch instruction returned is then written into the register specified by the returning tag identifier, and the instruction is then used when it is needed for an execution.

If the BTB value in the returning prefetch instruction and the current BTB value in BTR 19 are different, then this is an indication that a branch in the instruction stream has occurred. Consequently, the prefetch instruction is no longer needed and therefore, is not written in one of registers 15 through 18. The CFO bit associated with the returning prefetch instruction is then reset. At the time a branch in the instruction stream occurs, the CFO bits associated with each of registers 15 through 18 are examined. If any of these CFO bits are set, then the execution of the branch in the instruction stream must wait until all of these CFO bits are reset. Also at this time, the BTB in BTR 19 is complemented. Finally, control logic 21 examines each of registers 15 through 18 to determine if any prefetch has been sent to storage 14 but has not yet returned. If this is the case, then the CFO bit associated with that register is set thus indicating that there is a cancelled fetch outstanding.

The flow chart in Fig. 2 details the operation of IPB 11 at the time a branch in the instruction stream occurs. As indicated in box 22, a determination is made as to whether or not there are any cancelled fetches outstanding. If there are, then box 23 indicates that no further operation occurs until the cancelled tag identifying a returned prefetch instruction is sent from storage 14 to IPB 11. If there are no current fetches outstanding, then box 24 provides that control logic 21 sets a CFO bit for all registers 15 through 18 for which a prefetch has been allocated. Also at this time, the BTB value in BTR 19 is complemented as indicated in box 25 and all tags are set to their available state for enabling instructions to be fetched from storage as indicated in box 26. Finally in box 27, an identifying tag is selected in order to fetch the branch target instruction from storage 14. This prefetching process continues until all target fetches are completed.

The flow chart in Fig. 3 indicates the operation of IPB 11 when instruction execution unit 37 asks for the next instruction. As indicated in box 31, the next instruction address is calculated within IPB 11. Then in box 32, a decision is made as to whether or not the location in IPB 11 of the next instruction has been freed. If the answer is affirmative, then the freed IPB tag is set to its available state. However, if the answer is negative, then a further decision is made in block 34 as to whether or not the next instruction tag is in a valid state. If the next instruction tag is not in a valid state, then IPB 11 must wait for the selected tag to return from storage 14 over channel 12. However, if the next instruction tag is in a valid state, then the next instruction is provided to execution unit 37 for execution.

The process for selecting the next identifying tag for a prefetch is detailed in the flow chart shown in Fig. 4. After the next prefetch tag has been selected, a determination is made as to whether or not the tag is available. If such tag is not available, then IPB 11 must wait for such tag to become available as indicated in blocks 42 and 43. However, if the tag is available, then it is allocated to the next prefetch as shown in block 44. Finally the next prefetch along with identifying tag is sent to storage 14 so as to obtain the next instruction.

As shown in Fig. 5, when an instruction is returned from storage 14 with its associated identifying tag, a determination is made as to whether or not the BTB returning with the instruction is equal to the current BTB value in IPB 11. If the two values are not equal, then this is an indication that a branch has occurred in the instruction stream and consequently the prefetched instruction is no longer needed. The CFO bit for each returning tag is then reset. However, if the returning BTB value is equal to the current BTB value in IPB 11, then as stated previously this is an indication that a branch in the instruction stream has not occurred. A determination is then made as shown in block 53, as to whether or not a reply to an exception state exists. If the answer is affirmative then the tag identifier is set to the exception state, while if the answer is negative then the tag is set to the valid state and the instruction awaits execution.

Claims

1. A method for prefetching and executing a plurality of instructions having at least one branch instruction in a pipelined system comprising an instruction prefetch buffer (11) having a plurality of prefetch registers (15-18), a control logic (21) for directing the fetching of instructions to and from said prefetch registers, a storage (14), and a storage channel (12) for the exchange of instructions between said registers and said storage ;

   said method being characterized in that it comprises the steps of :

   generating a control tag (BTB) from said control logic each of whose current value is changed when a successful branch is execut-

* C L M S *

ed,

appending said control tag (BTB) to each request for instruction sent to said storage before said request is sent,

sending requests for instructions appended with said control tag (BTB) from said prefetch buffer to said storage,

receiving requested instructions from said storage along with said requests appended with said control tag (BTB),

determining if the value of the control tag appended to a request for a received instruction is equal to the current value of the control tag, and

retaining for execution each instruction received from said storage if the value of the appended control tag (BTB) is equal to the current value of the control tag and otherwise if the tags are unequal, ignoring the returning instruction.

2. The method according to Claim 1 further comprising the step of appending a number to each instruction request for identifying one of said plurality of prefetch registers (15-18) to which an instruction received from said storage (14) is to be sent.

3. The method according to Claim 1 or 2 further comprising the step of setting a control bit (CFO) for each of said plurality of prefetch registers (15-18) each time a request for instruction is sent to said storage (14) for that register.

4. The method according to Claim 3 further comprising the step of resetting said control bit (CFO) subsequent to the value of control tag accompanying each received instruction being unequal to the current value of said control tag, this inequality indicating that a branch instruction has been successfully executed.

5. The method according to claim 4 wherein the execution of the branch instruction is delayed until all control bits (CFO) for all said registers have been reset.

6. Pipelined system wherein a plurality of instructions having at least one Branch instruction are prefetched and executed, comprising an instruction prefetch buffer (11) having a plurality of prefetch registers (15-18), a control logic (21) for directing the fetching of instructions to and from said prefetch registers, a storage (14), and a storage channel (12) for the exchange of instructions between the prefetch registers and the storage ;

said system being characterized in that it comprises:

means for generating a control tag (BTB) from said control logic each of whose current value is changed when a successful branch is executed,

means for appending said control tag (BTB) to each request for instruction sent to said storage before said request is sent,

means for sending requests for instructions appended with said control tag (BTB) from said prefetch buffer to said storage,

means for receiving requested instructions from said storage along with said requests appended with said control tag (BTB),

means for determining if the value of the control tag appended to a request for a received instruction is equal to the current value of the control tag, and

means for retaining for execution each instruction received from said storage if the value of the appended control tag (BTB) is equal to the current value of the control tag and otherwise if the tags are unequal, ignoring the returning instruction.

Revendications

1. Procédé de préextraction et d'exécution d'une pluralité d'instructions comportant au moins une instruction de branchement dans un système de traitement pipeline comprenant un tampon (11) de préextraction d'instructions possédant une pluralité de registres de préextraction (15-18), une logique de commande (21) pour diriger l'extraction et l'envoi d'instructions vers et depuis les registres de préextraction, une mémoire (14) et un canal de mémoire (12) pour l'échange d'instructions entre les registres et la mémoire;
procédé qui est caractérisé en ce qu'il comprend les étapes consistant à:
générer une étiquette de contrôle (BTB) à partir de la logique de commande, étiquette dont la valeur actuelle est changée chaque fois qu'un branchement est effectivement exécuté,

* C L M S *

annexer cette étiquette de contrôle (BTB) à chaque demande d'instruction envoyée à la mémoire, l'annexation étant effectuée avant l'envoi de la demande,
envoyer des demandes d'instructions, auxquelles sont annexées des étiquettes de contrôle (BTB), du tampon de préextraction à la mémoire,
recevoir les instructions demandées en provenance de la mémoire, ensemble avec les demandes auxquelles sont annexées les étiquettes de contrôle (BTB),
déterminer si la valeur de l'étiquette de contrôle annexée à une demande pour une instruction reçue est égale à la valeur actuelle de l'étiquette et
retenir, en vue de l'exécution, chaque instruction reçue de la mémoire si la valeur de l'étiquette de contrôle (BTB) annexée est égale à la valeur actuelle de l'étiquette de contrôle sinon, au cas où les étiquettes sont différentes, ignorer l'instruction retournée.

2. Procédé selon la revendication 1, comprenant en outre l'étape consistant à annexer à chaque demande d'instruction un numéro identifiant celui des registres de préextraction (15-18) auquel doit être envoyée l'instruction reçue de la mémoire (14).

3. Procédé selon la revendication 1 ou 2, comprenant en outre l'étape consistant à positionner un bit de contrôle (CFO) pour chacun des registres de préextraction (15-18) chaque fois qu'une demande d'instruction est envoyée à la mémoire (14) pour le registre en question.

4. Procédé selon la revendication 3, comprenant en outre l'étape consistant à mettre à zéro le bit de contrôle (CFO) après qu'il a été constaté que la valeur de l'étiquette de contrôle accompagnant chaque instruction reçue diffère de la valeur actuelle de l'étiquette de contrôle, cette inégalité indiquant qu'une instruction de branchement a effectivement été exécutée.

5. Procédé selon la revendication 4, dans lequel l'exécution de l'instruction de branchement est retardée jusqu'à ce que tous les bits de contrôle (CFO) pour tous les registres aient été mis à zéro.

6. Système de traitement pipeline dans lequel sont préextraites et exécutées une pluralité d'instructions comportant au moins une instruction de branchement, comprenant un tampon (11) de préextraction d'instructions possédant une pluralité de registres de préextraction (15-18), une logique de commande (21) pour diriger l'extraction et l'envoi d'instructions vers et depuis les registres de préextraction, une mémoire (14) et un canal de mémoire (12) pour l'échange d'instructions entre les registres et la mémoire;
système qui est caractérisé en ce qu'il comprend:
un moyen pour générer une étiquette de contrôle (BTB) à partir de la logique de commande, étiquette dont la valeur actuelle est changée lorsqu'un branchement est effectivement exécuté,
un moyen pour annexer l'étiquette de contrôle (BTB) à chaque demande d'instruction envoyée à la mémoire, l'annexation étant effectuée avant l'envoi de la demande,
un moyen pour envoyer des demandes d'instructions, auxquelles sont annexées des étiquettes de contrôle (BTB), du tampon de préextraction à la mémoire,
un moyen pour recevoir les instructions demandées en provenance de la mémoire, ensemble avec les demandes auxquelles sont annexées les étiquettes de contrôle (BTB),
un moyen pour déterminer si la valeur de l'étiquette de contrôle annexée à une demande pour une instruction reçue est égale à la valeur actuelle de l'étiquette de contrôle et
un moyen pour retenir, en vue de l'exécution, chaque instruction reçue de la mémoire si la valeur de l'étiquette de contrôle (BTB) annexée est égale à la valeur actuelle de l'étiquette de contrôle sinon, au cas où les étiquettes sont différentes, ignorer l'instruction retournée.

Patentansprüche

1. Verfahren zum Vorauslesen und zum Ausführen einer Mehrzahl von Befehlen mit zumindest einem Verzweigungsbefehl in einem System, das einem Pipelining unterworfen ist, mit einem Vorauslesepuffer (11), der folgendes aufweist: eine Mehrzahl von Vorausleseregistern (15-18), eine Steuerlogik (21) zum Lenken des Auslesens von Befehlen an die Vorausleseregister und aus denselben, einen Speicher (14) und einen Speicherkanal (12) für den Austausch von Befehlen zwischen den Registern und dem Speicher,

wobei das Verfahren dadurch gekennzeichnet ist, daß dieses die folgenden Schritte umfaßt:

Erzeugen eines Steuerkennzeichens (BTB) aus der Steuerlogik, wobei jedes einen laufenden Wert aufweist, der geändert wird, wenn eine

* C L M S *

erfolgreiche Verzweigung ausgeführt wird,

Anfügen des Steuerkennzeichens (BTB) jeder Anforderung für einen Befehl, der an den Speicher gesendet ist, bevor die Anforderung gesendet wird,

Senden von Anforderungen für Befehle, welchen das Steuerkennzeichen (BTB) angefügt ist, aus dem Vorauslesepuffer zu dem Speicher,

Empfangen angeforderter Befehle aus dem Speicher zusammen mit den Anforderungen, welchen das Steuerkennzeichen (BTB) angefügt ist,

Bestimmen, ob der Wert des Steuerzeichens, das einer Anforderung für einen empfangenen Befehl angefügt ist, gleich dem laufenden Wert des Steuerkennzeichens ist und

Zurückbehalten jedes Befehls für eine Ausführung, der aus dem Speicher empfangen ist, falls der Wert des angefügten Steuerzeichens (BTB) gleich dem laufenden Wert des Steuerzeichens ist und andernfalls, falls die Kennzeichen ungleich sind, Ignorieren des zurückkehrenden Befehls.

2. Verfahren nach Anspruch 1, das ferner den Schritt des Anfügens einer Zahl jeder Befehlsanforderung für Identifizierung eines aus der Mehrzahl von Vorausleseregistern (15-18) umfaßt, an die ein aus dem Speicher (14) empfangener Befehl gesendet werden soll.

3. Verfahren nach Anspruch 1 oder 2, das ferner den folgenden Schritt umfaßt: Setzen eines Steuerbits (CFO) für jedes aus der Mehrzahl von Vorausleseregistern (15-18) jedes Mal, wenn eine Anforderung für einen Befehl an den Speicher (14) für jenes Register gesendet wird.

4. Verfahren nach Anspruch 3, das ferner den folgenden Schritt umfaßt: Zurücksetzen des Steuerbits (CFO) dem folgend, daß der Wert des Steuerzeichens, das jeden empfangenen Befehl begleitet, ungleich dem laufenden Wert des Steuerzeichens ist, wobei diese Ungleichheit anzeigt, daß ein Verzweigungsbefehl erfolgreich ausgeführt wurde.

5. Verfahren nach Anspruch 4, bei welchem das Ausführen des Verzweigungsbefehls solange verzögert wird, bis alle Steuerbits (CFO) für alle Register zurückgesetzt wurden.

6. Einem Pipelining unterworfenes System, bei welchem eine Mehrzahl von Befehlen mit zumindest einem Verzweigungsbefehl vorausgelesen und ausgeführt wird, mit einem Vorauslesepuffer (11), der folgendes aufweist: eine Mehrzahl von Vorausleseregistern (15-18), eine Steuerlogik (21) zum Lenken des Auslesens von Befehlen an die Vorausleseregister und aus denselben, einen Speicher (14) und einen Speicherkanal (12) für den Austausch von Befehlen zwischen den Registern und dem Speicher,

wobei das System dadurch gekennzeichnet ist, daß dieses folgendes aufweist:

Mittel zum Erzeugen eines Steuerkennzeichens (BTB) aus der Steuerlogik, wobei jedes einen laufenden Wert aufweist, der geändert wird, wenn eine erfolgreiche Verzweigung ausgeführt wird,

Mittel zum Anfügen des Steuerkennzeichens (BTB) jeder Anforderung für einen Befehl, der an den Speicher gesendet ist, bevor die Anforderung gesendet wird,

Mittel zum Senden von Anforderungen für Befehle, welchen das Steuerkennzeichen (BTB) angefügt ist, aus dem Vorauslesepuffer zu dem Speicher,

Mittel zum Empfangen angeforderter Befehle aus dem Speicher zusammen mit den Anforderungen, welchen das Steuerkennzeichen (BTB) angefügt ist,

Mittel zum Bestimmen, ob der Wert des Steuerzeichens, das einer Anforderung für einen empfangenen Befehl angefügt ist, gleich dem laufenden Wert des Steuerkennzeichens ist und

Mittel zum Zurückbehalten jedes Befehls für eine Ausführung, der aus dem Speicher empfangen ist, falls der Wert des angefügten Steuerzeichens (BTB) gleich dem laufenden Wert des Steuerzeichens ist und andernfalls, falls die Kennzeichen ungleich sind, zum Ignorieren des zurückkehrenden Befehls.