Is branch prediction important for performance?

Daniel J. Bernstein

Spectre paper: “Modern processors use branch prediction and speculative execution to maximize performance.”

Wikipedia: “Branch predictors play a critical role in achieving high effective performance in many modern pipelined microprocessor architectures such as x86.”

The article cited by Wikipedia says: “Branch predictor (BP) is an essential component in modern processors since high BP accuracy can improve performance and reduce energy by decreasing the number of instructions executed on wrong-path.”
Is branch prediction important for performance?

Daniel J. Bernstein

Spectre paper: “Modern processors use branch prediction and speculative execution to maximize performance.”

Wikipedia: “Branch predictors play a critical role in achieving high effective performance in many modern pipelined microprocessor architectures such as x86.”

The article cited by Wikipedia says: “Branch predictor (BP) is an essential component in modern processors since high BP accuracy can improve performance and reduce energy by decreasing the number of instructions executed on wrong-path.”

— Omitting branch prediction reduces energy even more. Eliminates all wrong-path instructions. Also eliminates cost of prediction + speculation.
Is branch prediction important for performance?

Daniel J. Bernstein

Spectre paper: “Modern processors use branch prediction and speculative execution to maximize performance.”

Wikipedia: “Branch predictors play a critical role in achieving high effective performance in many modern pipelined microprocessor architectures such as x86.”

The article cited by Wikipedia says: “Branch predictor (BP) is an essential component in modern processors since high BP accuracy can improve performance and reduce energy by decreasing the number of instructions executed on wrong-path.”

— Omitting branch prediction reduces energy even more. Eliminates all wrong-path instructions. Also eliminates cost of prediction+speculation.

The real question is **latency**.
Is branch prediction important for performance?

Daniel J. Bernstein

Spectre paper: “Modern processors use branch prediction and speculative execution to maximize performance.”

Wikipedia: “Branch predictors play a critical role in achieving high effective performance in many modern pipelined microprocessor architectures such as x86.”

The article cited by Wikipedia says: “Branch predictor (BP) is an essential component in modern processors since high BP accuracy can improve performance and reduce energy by decreasing the number of instructions executed on wrong-path.”

— Omitting branch prediction reduces energy even more. Eliminates all wrong-path instructions. Also eliminates cost of prediction+speculation.

The real question is latency.

The CPU pipeline
Cycle 1:

<table>
<thead>
<tr>
<th>fetch</th>
<th>decode</th>
<th>register read</th>
<th>execute</th>
<th>register write</th>
</tr>
</thead>
</table>


Is branch prediction important for performance?

Daniel J. Bernstein

Spectre paper: “Modern processors use branch prediction and speculative execution to maximize performance.”

Wikipedia: “Branch predictors play a critical role in achieving high effective performance in many modern pipelined microprocessor architectures such as x86.”

The article cited by Wikipedia says: “Branch predictor (BP) is an essential component in modern processors since high BP accuracy can improve performance and reduce energy by decreasing the number of instructions executed on wrong-path.”

— Omitting branch prediction reduces energy even more. Eliminates all wrong-path instructions. Also eliminates cost of prediction+speculation.

The real question is latency.
Is branch prediction important for performance?

Daniel J. Bernstein

Spectre paper: “Modern processors use branch prediction and speculative execution to maximize performance.”

Wikipedia: “Branch predictors play a critical role in achieving high effective performance in many modern pipelined microprocessor architectures such as x86.”

The article cited by Wikipedia says: “Branch predictor (BP) is an essential component in modern processors since high BP accuracy can improve performance and reduce energy by decreasing the number of instructions executed on wrong-path.”

— Omitting branch prediction reduces energy even more. Eliminates all wrong-path instructions. Also eliminates cost of prediction+speculation.

The real question is latency.

The CPU pipeline

Cycle 1:

<table>
<thead>
<tr>
<th>fetch</th>
<th>a=b+c</th>
</tr>
</thead>
<tbody>
<tr>
<td>decode</td>
<td></td>
</tr>
<tr>
<td>register read</td>
<td></td>
</tr>
<tr>
<td>execute</td>
<td></td>
</tr>
<tr>
<td>register write</td>
<td></td>
</tr>
</tbody>
</table>
The article cited by Wikipedia says: “Branch predictor (BP) is an essential component in modern processors since high BP accuracy can improve performance and reduce energy by decreasing the number of instructions executed on wrong-path.”

— Omitting branch prediction reduces energy even more. Eliminates all wrong-path instructions. Also eliminates cost of prediction+speculation.

The real question is **latency**.
The article cited by Wikipedia says: “Branch predictor (BP) is an essential component in modern processors since high BP accuracy can improve performance and reduce energy by decreasing the number of instructions executed on wrong-path.”

— Omitting branch prediction reduces energy even more. Eliminates all wrong-path instructions. Also eliminates cost of prediction+speculation.

The real question is latency.
The article cited by Wikipedia says: “Branch predictor (BP) is an essential component in modern processors since high BP accuracy can improve performance and reduce energy by decreasing the number of instructions executed on wrong-path.”

— Omitting branch prediction reduces energy even more. Eliminates all wrong-path instructions. Also eliminates cost of prediction + speculation.

The real question is latency.
The article cited by Wikipedia says: “Branch predictor (BP) is an essential component in modern processors since high BP accuracy can improve performance and reduce energy by decreasing the number of instructions executed on wrong-path.”

— Omitting branch prediction reduces energy even more. Eliminates all wrong-path instructions. Also eliminates cost of prediction+speculation.

The real question is latency.
The article cited by Wikipedia says: “Branch predictor (BP) is an essential component in modern processors since high BP accuracy can improve performance and reduce energy by decreasing the number of instructions executed on wrong-path.”

— Omitting branch prediction reduces energy even more. Eliminates all wrong-path instructions. Also eliminates cost of prediction+speculation.

The real question is **latency**.
The article cited by Wikipedia says: “Branch predictor (BP) is an essential component in modern processors since high BP accuracy can improve performance and reduce energy by decreasing the number of instructions executed on wrong-path.”

— Omitting branch prediction reduces energy even more. Eliminates all wrong-path instructions. Also eliminates cost of prediction+speculation.

The real question is **latency**.
The article cited by Wikipedia says: “Branch predictor (BP) is an essential component in modern processors since high BP accuracy can improve performance and reduce energy by decreasing the number of instructions executed on wrong-path.”

— Omitting branch prediction reduces energy even more. Eliminates all wrong-path instructions. Also eliminates cost of prediction+speculation.

The real question is latency.

The CPU pipeline
Cycle 2:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>fetch</td>
<td>( d = e + f )</td>
</tr>
<tr>
<td>decode</td>
<td>( a = b + c )</td>
</tr>
<tr>
<td>register read</td>
<td></td>
</tr>
<tr>
<td>execute</td>
<td></td>
</tr>
<tr>
<td>register write</td>
<td></td>
</tr>
</tbody>
</table>

Second instruction is fetched; first instruction is decoded. Hardware units operate in parallel.
The article cited by Wikipedia says: “Branch predictor (BP) is an essential component in modern processors since high BP accuracy can improve performance and reduce energy by decreasing the number of instructions executed on wrong-path.”

— Omitting branch prediction reduces energy even more. Eliminates all wrong-path instructions. Also eliminates cost of prediction+speculation.

The real question is latency.
The article cited by Wikipedia says: “Branch predictor (BP) is an essential component in modern processors since high BP accuracy can improve performance and reduce energy by decreasing the number of instructions executed on wrong-path.”

— Omitting branch prediction reduces energy even more. Eliminates all wrong-path instructions. Also eliminates cost of prediction+speculation.

The real question is latency.
The article cited by Wikipedia says: “Branch predictor (BP) is an essential component in modern processors since high BP accuracy can improve performance and reduce energy by decreasing the number of instructions executed on wrong-path.”

— Omitting branch prediction reduces energy even more. Eliminates all wrong-path instructions. Also eliminates cost of prediction+speculation.

The real question is latency.
The article cited by Wikipedia says: “Branch predictor (BP) is an essential component in modern processors since high BP accuracy can improve performance and reduce energy by decreasing the number of instructions executed on wrong-path.”

— Omitting branch prediction reduces energy even more. Eliminates all wrong-path instructions. Also eliminates cost of prediction+speculation.

The real question is latency.
The article cited by Wikipedia says: “Branch predictor (BP) is an essential component in modern processors since high BP accuracy can improve performance and reduce energy by decreasing the number of instructions executed on wrong-path.”

— Omitting branch prediction reduces energy even more. Eliminates all wrong-path instructions. Also eliminates cost of prediction+speculation.

The real question is latency.
The article cited by Wikipedia says: “Branch predictor (BP) is an essential component in modern processors since high BP accuracy can improve performance and reduce energy by decreasing the number of instructions executed on wrong-path.”

— Omitting branch prediction reduces energy even more. Eliminates all wrong-path instructions. Also eliminates cost of prediction+speculation.

The real question is latency.
The article cited by Wikipedia says: “Branch predictor (BP) is an essential component in modern processors since high BP accuracy can improve performance and reduce energy by decreasing the number of instructions executed on wrong-path.”

— Omitting branch prediction reduces energy even more. Eliminates all wrong-path instructions. Also eliminates cost of prediction+speculation.

The real question is latency.
The article cited by Wikipedia says: “Branch predictor (BP) is an essential component in modern processors since high BP accuracy can improve performance and reduce energy by decreasing the number of instructions executed on wrong-path.”

— Omitting branch prediction reduces energy even more. Eliminates all wrong-path instructions. Also eliminates cost of prediction + speculation.

The real question is **latency**.
The article cited by Wikipedia says: “Branch predictor (BP) is an essential component in modern processors since high BP accuracy can improve performance and reduce energy by decreasing the number of instructions executed on wrong-path.”

— Omitting branch prediction reduces energy even more. Eliminates all wrong-path instructions. Also eliminates cost of prediction+speculation.

The real question is **latency**.
The article cited by Wikipedia says: “Branch predictor (BP) is an essential component in modern processors since high BP accuracy can improve performance and reduce energy by decreasing the number of instructions executed on wrong-path.”

— Omitting branch prediction reduces energy even more. Eliminates all wrong-path instructions. Also eliminates cost of prediction+speculation.

The real question is **latency**.
The article cited by Wikipedia says: “Branch predictor (BP) is an essential component in modern processors since high BP accuracy can improve performance and reduce energy by decreasing the number of instructions executed on wrong-path.”

— Omitting branch prediction reduces energy even more. Eliminates all wrong-path instructions. Also eliminates cost of prediction+speculation.

The real question is **latency**.
The article cited by Wikipedia says: “Branch predictor (BP) is an essential component in modern processors since high BP accuracy can improve performance and reduce energy by decreasing the number of instructions executed on wrong-path.”

— Omitting branch prediction reduces energy even more. Eliminates all wrong-path instructions. Also eliminates cost of prediction+speculation.

The real question is latency.
The article cited by Wikipedia says: “Branch predictor (BP) is an essential component in modern processors since high BP accuracy can improve performance and reduce energy by decreasing the number of instructions executed on wrong-path.”

— Omitting branch prediction reduces energy even more. Eliminates all wrong-path instructions. Also eliminates cost of prediction+speculation.

The real question is latency.

<table>
<thead>
<tr>
<th>CPU pipeline</th>
<th>Cycle 5:</th>
</tr>
</thead>
<tbody>
<tr>
<td>fetch</td>
<td></td>
</tr>
<tr>
<td>decode</td>
<td>if(g&lt;0)</td>
</tr>
<tr>
<td>register read</td>
<td>g=h-i</td>
</tr>
<tr>
<td>execute</td>
<td>d=e+f</td>
</tr>
<tr>
<td>register write</td>
<td>a=b+c</td>
</tr>
</tbody>
</table>

Without branch prediction, fetch unit doesn’t know which instruction to fetch now! Waiting for if to write “instruction pointer” register.
The article cited by Wikipedia says: “Branch predictor (BP) is an essential component in modern processors since high BP accuracy can improve performance and reduce energy by decreasing the number of instructions executed on wrong-path.”

— Omitting branch prediction reduces energy even more. Eliminates all wrong-path instructions. Also eliminates cost of prediction+speculation.

The real question is latency.

The CPU pipeline

Cycle 6:

<table>
<thead>
<tr>
<th>Stage</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>fetch</td>
<td></td>
</tr>
<tr>
<td>decode</td>
<td></td>
</tr>
<tr>
<td>register read</td>
<td>if(g&lt;0)</td>
</tr>
<tr>
<td>execute</td>
<td>g=h-i</td>
</tr>
<tr>
<td>register write</td>
<td>d=e+f</td>
</tr>
</tbody>
</table>

Fetch is still waiting.

Typical CPUs: longer pipelines; longer delays than this picture.

(Assume no hyperthreading.)
The article cited by Wikipedia says: “Branch predictor (BP) is an essential component in modern processors since high BP accuracy can improve performance and reduce energy by decreasing the number of instructions executed on wrong-path.”

— Omitting branch prediction reduces energy even more. Eliminates all wrong-path instructions. Also eliminates cost of prediction+speculation.

The real question is **latency**.
The article cited by Wikipedia says: “Branch predictor (BP) is an essential component in modern processors since high BP accuracy can improve performance and reduce energy by decreasing the number of instructions executed on wrong-path.”

— Omitting branch prediction reduces energy even more. Eliminates all wrong-path instructions. Also eliminates cost of prediction+speculation.

The real question is **latency**.
The article cited by Wikipedia says: “Branch predictor (BP) is an essential component in modern processors since high BP accuracy can improve performance and reduce energy by decreasing the number of instructions executed on wrong-path.”

— Omitting branch prediction reduces energy even more. Eliminates all wrong-path instructions. Also eliminates cost of prediction+speculation.

The real question is **latency**.
The article cited by Wikipedia says: “Branch predictor (BP) is an essential component in modern processors since high BP accuracy can improve performance and reduce energy by decreasing the number of instructions executed on wrong-path.”

— Omitting branch prediction reduces energy even more. Eliminates all wrong-path instructions. Also eliminates cost of prediction+speculation.

The real question is latency.
The article cited by Wikipedia says: “Branch predictor (BP) is an essential component in modern processors since high BP accuracy can improve performance and reduce energy by decreasing the number of instructions executed on wrong-path.”

— Omitting branch prediction reduces energy even more. Eliminates all wrong-path instructions. Also eliminates cost of prediction+speculation.

The real question is latency.
The article cited by Wikipedia says: “Branch predictor (BP) is an essential component in modern processors since high BP accuracy can improve performance and reduce energy by decreasing the number of instructions executed on wrong-path.”

— Omitting branch prediction reduces energy even more. Eliminates all wrong-path instructions. Also eliminates cost of prediction+speculation.

The real question is latency.

---

The CPU pipeline

Cycle 5:

<table>
<thead>
<tr>
<th>fetch</th>
<th>if(?)</th>
</tr>
</thead>
<tbody>
<tr>
<td>decode</td>
<td>j=k+l</td>
</tr>
<tr>
<td>register read</td>
<td>d=e+f</td>
</tr>
<tr>
<td>execute</td>
<td>a=b+c</td>
</tr>
<tr>
<td>register write</td>
<td>&lt;0? g=h-i</td>
</tr>
</tbody>
</table>

Fast-forward flag to fetch unit. Branch prediction has zero benefit if programs compute branch conditions \( P \) cycles in advance, where \( P \) is pipeline length.
The article cited by Wikipedia says: “Branch predictor (BP) is an essential component in modern processors since high BP accuracy improves performance and energy by decreasing the number of instructions executed on wrong-path.”

Omitting branch prediction reduces energy even more. Omitting branch prediction also eliminates all wrong-path instructions. Also eliminates cost of prediction + speculation. The real question is latency.

The CPU pipeline

Cycle 5:

<table>
<thead>
<tr>
<th>Stage</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>fetch</td>
<td>if(?)</td>
</tr>
<tr>
<td>decode</td>
<td>j=k+l</td>
</tr>
<tr>
<td>register read</td>
<td>d=e+f</td>
</tr>
<tr>
<td>execute</td>
<td>a=b+c</td>
</tr>
<tr>
<td>register write</td>
<td>&lt;0? g=h−i</td>
</tr>
</tbody>
</table>

Fast-forward flag to fetch unit. Branch prediction has zero benefit if programs compute branch conditions $P$ cycles in advance, where $P$ is pipeline length.

CPUs today spend almost all time applying simple computations to large volumes of data. Massively parallelizable.

Why shouldn’t programs compute branch conditions in advance?
The article cited by Wikipedia says: "Branch predictor (BP) is an essential component in modern processors since high BP accuracy improves performance and decreasing the number of instructions on wrong-path can improve performance and reduce energy by decreasing the number of instructions executed on wrong-path."

Omitting branch prediction can reduce energy even more. Eliminates all wrong-path instructions. Also eliminates cost of prediction + speculation.

The real question is latency.

CPUs today spend almost all time applying simple computations to large volumes of data. Massively parallelizable. Why shouldn't programs compute branch conditions in advance, where $P$ is pipeline length?

The CPU pipeline

<table>
<thead>
<tr>
<th>Cycle 5:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>fetch</td>
<td>if(?)</td>
</tr>
<tr>
<td>decode</td>
<td>$j = k + l$</td>
</tr>
<tr>
<td>register read</td>
<td>$d = e + f$</td>
</tr>
<tr>
<td>execute</td>
<td>$a = b + c$</td>
</tr>
<tr>
<td>register write</td>
<td>$&lt; 0? \ g = h - i$</td>
</tr>
</tbody>
</table>

Fast-forward flag to fetch unit.

Branch prediction has zero benefit if programs compute branch conditions $P$ cycles in advance, where $P$ is pipeline length.
The article cited by Wikipedia says: “Branch predictor (BP) is an essential component in modern processors since high BP accuracy can improve performance and reduce energy by decreasing the number of instructions executed on wrong-path. Omitting branch prediction reduces energy even more. Eliminates all wrong-path instructions. Also eliminates cost of prediction+speculation. The real question is latency.

The CPU pipeline

Cycle 5:

<table>
<thead>
<tr>
<th>Fetch</th>
<th>if(?)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>j=k+l</td>
</tr>
<tr>
<td>Register Read</td>
<td>d=e+f</td>
</tr>
<tr>
<td>Execute</td>
<td>a=b+c</td>
</tr>
<tr>
<td>Register Write</td>
<td>&lt;0? g=h-i</td>
</tr>
</tbody>
</table>

Fast-forward flag to fetch unit. Branch prediction has zero benefit if programs compute branch conditions \( P \) cycles in advance, where \( P \) is pipeline length.

CPUs today spend almost all time applying simple computations to large volumes of data. Massively parallelizable. Why shouldn’t programs compute branch conditions in advance?
The CPU pipeline

Cycle 5:

<table>
<thead>
<tr>
<th>Stage</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>fetch</td>
<td>if(?)</td>
</tr>
<tr>
<td>decode</td>
<td>j=k+l</td>
</tr>
<tr>
<td>register read</td>
<td>d=e+f</td>
</tr>
<tr>
<td>execute</td>
<td>a=b+c</td>
</tr>
<tr>
<td>register write</td>
<td>&lt;0?     g=h-i</td>
</tr>
</tbody>
</table>

Fast-forward flag to fetch unit.
Branch prediction has zero benefit if programs compute branch conditions \( P \) cycles in advance, where \( P \) is pipeline length.

CPUs today spend almost all time applying simple computations to large volumes of data. Massively parallelizable.

Why shouldn’t programs compute branch conditions in advance?
The CPU pipeline

Cycle 5:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>fetch</td>
<td>if(?)</td>
</tr>
<tr>
<td>decode</td>
<td>j=k+l</td>
</tr>
<tr>
<td>register read</td>
<td>d=e+f</td>
</tr>
<tr>
<td>execute</td>
<td>a=b+c</td>
</tr>
<tr>
<td>register write</td>
<td>&lt;0? g=h−i</td>
</tr>
</tbody>
</table>

Fast-forward flag to fetch unit.
Branch prediction has zero benefit if programs compute branch conditions \( P \) cycles in advance, where \( P \) is pipeline length.

CPUs today spend almost all time applying simple computations to large volumes of data. Massively parallelizable.

Why shouldn’t programs compute branch conditions in advance?
Most cases are handled by simple instruction scheduling.
The CPU pipeline

Cycle 5:

<table>
<thead>
<tr>
<th>Step</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>fetch</td>
<td>if(?)</td>
</tr>
<tr>
<td>decode</td>
<td>j=k+l</td>
</tr>
<tr>
<td>register read</td>
<td>d=e+f</td>
</tr>
<tr>
<td>execute</td>
<td>a=b+c</td>
</tr>
<tr>
<td>register write</td>
<td>&lt;0? g=h–i</td>
</tr>
</tbody>
</table>

Fast-forward flag to fetch unit.
Branch prediction has zero benefit
**if programs compute branch conditions** $P$ **cycles in advance**, where $P$ is pipeline length.

CPUs today spend almost all time applying simple computations to large volumes of data. Massively parallelizable.

Why shouldn’t programs compute branch conditions in advance?

Most cases are handled by simple instruction scheduling.

Insn-set extensions for more cases: “branch-relevant” priority bit; multiple flags; loop counter. (Count down early in pipeline.)

Inner loops I’ve studied don’t need more complicated patterns.
The CPU pipeline

Cycle 5:
fetch if(?)
decode j=k+l
register read d=e+f
execute a=b+c
register write <0? g=h-i

Fast-forward flag to fetch unit.
Branch prediction has zero benefit if programs compute branch conditions $P$ cycles in advance, where $P$ is pipeline length.

CPUs today spend almost all time applying simple computations to large volumes of data. Massively parallelizable.

Why shouldn’t programs compute branch conditions in advance?

Most cases are handled by simple instruction scheduling.

Insn-set extensions for more cases: “branch-relevant” priority bit; multiple flags; loop counter. (Count down early in pipeline.) Inner loops I’ve studied don’t need more complicated patterns.

How did the community convince itself that branch prediction is important for performance?
The CPU pipeline:

Cycle 5:
- Fetch if(?)
- Decode j=k+l
- Register read d=e+f
- Execute a=b+c
- Register write <0? g=h-i

Fast-forward flag to fetch unit.

Branch prediction has zero benefit if programs compute branch conditions \( P \) cycles in advance, where \( P \) is pipeline length.

CPUs today spend almost all time applying simple computations to large volumes of data. Massively parallelizable.

Why shouldn’t programs compute branch conditions in advance?

Most cases are handled by simple instruction scheduling.

Insn-set extensions for more cases: “branch-relevant” priority bit; multiple flags; loop counter. (Count down early in pipeline.) Inner loops I’ve studied don’t need more complicated patterns.

How did the community convince itself that branch prediction is important for performance?
CPUs today spend almost all time applying simple computations to large volumes of data. Massively parallelizable.

Why shouldn’t programs compute branch conditions in advance?

Most cases are handled by simple instruction scheduling.

Insn-set extensions for more cases: “branch-relevant” priority bit; multiple flags; loop counter.

(Count down early in pipeline.) Inner loops I’ve studied don’t need more complicated patterns.

How did the community convince itself that branch prediction is important for performance?
CPUs today spend almost all time applying simple computations to large volumes of data. Massively parallelizable.

Why shouldn’t programs compute branch conditions in advance?

Most cases are handled by simple instruction scheduling.

Insn-set extensions for more cases: “branch-relevant” priority bit; multiple flags; loop counter. (Count down early in pipeline.)

Inner loops I’ve studied don’t need more complicated patterns.

How did the community convince itself that branch prediction is important for performance?
CPUs today spend almost all time applying simple computations to large volumes of data. Massively parallelizable.

Why shouldn't programs compute branch conditions in advance?

Most cases are handled by simple instruction scheduling.

Insn-set extensions for more cases: “branch-relevant” priority bit; multiple flags; loop counter. (Count down early in pipeline.) Inner loops I’ve studied don’t need more complicated patterns.

How did the community convince itself that branch prediction is important for performance?

1980s insn sets, CPU costs → 1990s compilers, applications, data volumes, compiled code → 1990s/2000s hype (e.g., “Since programs typically encounter branches every 4–6 instructions, inaccurate branch prediction causes a severe performance degradation in highly superscalar or deeply pipelined designs”) → 2000s/2010s beliefs.
CPUs today spend almost all time applying simple computations to large volumes of data. They are massively parallelizable.

Why shouldn’t programs compute branch conditions in advance? Most cases are handled by instruction scheduling.

Insn-set extensions for more cases: “branch-relevant” priority bit; multiple flags; loop counter. (Count down early in pipeline.) Inner loops I’ve studied don’t need more complicated patterns.

How did the community convince itself that branch prediction is important for performance?

1980s insn sets, CPU costs → 1990s compilers, applications, data volumes, compiled code → 1990s/2000s hype (e.g., “Since programs typically encounter branches every 4–6 instructions, inaccurate branch prediction causes a severe performance degradation in highly superscalar or deeply pipelined designs”) → 2000s/2010s beliefs.

The fundamental question: Can a well-designed insn set with well-designed software remove the speed incentive for branch prediction?
CPUs today spend almost all time applying simple computations to large volumes of data. Massively parallelizable. Why shouldn’t programs compute branch conditions in advance? Most cases are handled by simple instruction scheduling. Insn-set extensions for more cases: “branch-relevant” priority bit; multiple flags; loop counter. (Count down early in pipeline.) Inner loops I’ve studied don’t need more complicated patterns.

How did the community convince itself that branch prediction is important for performance?
1980s insn sets, CPU costs → 1990s compilers, applications, data volumes, compiled code → 1990s/2000s hype (e.g., “Since programs typically encounter branches every 4–6 instructions, inaccurate branch prediction causes a severe performance degradation in highly superscalar or deeply pipelined designs”) → 2000s/2010s beliefs.

The fundamental question: Can a well-designed insn set with well-designed software remove the speed incentive for branch prediction?
CPUs today spend almost all time applying simple computations to large volumes of data. Massively parallelizable. Why shouldn't programs compute branch conditions in advance? Most cases are handled by simple instruction scheduling.Insn-set extensions for more cases: "branch-relevant" priority bit; multiple flags; loop counter. (Count down early in pipeline.) Inner loops I've studied don't need more complicated patterns.

How did the community convince itself that branch prediction is important for performance?

1980s insn sets, CPU costs → 1990s compilers, applications, data volumes, compiled code → 1990s/2000s hype (e.g., “Since programs typically encounter branches every 4–6 instructions, inaccurate branch prediction causes a severe performance degradation in highly superscalar or deeply pipelined designs”) → 2000s/2010s beliefs.

The fundamental question: Can a well-designed insn set with well-designed software remove the speed incentive for branch prediction?
How did the community convince itself that branch prediction is important for performance?

1980s insn sets, CPU costs \(\rightarrow\) 1990s compilers, applications, data volumes, compiled code \(\rightarrow\) 1990s/2000s hype (e.g., “Since programs typically encounter branches every 4–6 instructions, inaccurate branch prediction causes a severe performance degradation in highly superscalar or deeply pipelined designs”) \(\rightarrow\) 2000s/2010s beliefs.

The fundamental question: Can a well-designed insn set with well-designed software remove the speed incentive for branch prediction?
How did the community convince itself that branch prediction is important for performance?

1980s insn sets, CPU costs → 1990s compilers, applications, data volumes, compiled code → 1990s/2000s hype (e.g., “Since programs typically encounter branches every 4–6 instructions, inaccurate branch prediction causes a severe performance degradation in highly superscalar or deeply pipelined designs”) → 2000s/2010s beliefs.

The fundamental question: Can a well-designed insn set with well-designed software remove the speed incentive for branch prediction?

“We need to look at current insn sets.”
How did the community convince itself that branch prediction is important for performance?

1980s insn sets, CPU costs → 1990s compilers, applications, data volumes, compiled code → 1990s/2000s hype (e.g., “Since programs typically encounter branches every 4–6 instructions, inaccurate branch prediction causes a severe performance degradation in highly superscalar or deeply pipelined designs”) → 2000s/2010s beliefs.

The fundamental question: Can a well-designed insn set with well-designed software remove the speed incentive for branch prediction?

“We need to look at current insn sets.” — Yes, interesting short-term question. Not my question in this talk.
How did the community convince itself that branch prediction is important for performance?

1980s insn sets, CPU costs \rightarrow 1990s compilers, applications, data volumes, compiled code \rightarrow 1990s/2000s hype (e.g., “Since programs typically encounter branches every 4–6 instructions, inaccurate branch prediction causes a severe performance degradation in highly superscalar or deeply pipelined designs”) \rightarrow 2000s/2010s beliefs.

The fundamental question: Can a well-designed insn set with well-designed software remove the speed incentive for branch prediction?

“We need to look at current insn sets.” — Yes, interesting short-term question. Not my question in this talk.

“We need to look at badly written software.”
How did the community convince itself that branch prediction is important for performance?

1980s insn sets, CPU costs → 1990s compilers, applications, data volumes, compiled code → 1990s/2000s hype (e.g., “Since programs typically encounter branches every 4–6 instructions, inaccurate branch prediction causes a severe performance degradation in highly superscalar or deeply pipelined designs”) → 2000s/2010s beliefs.

The fundamental question: Can a well-designed insn set with well-designed software remove the speed incentive for branch prediction?

“We need to look at current insn sets.” — Yes, interesting short-term question. Not my question in this talk.

“We need to look at badly written software.” — No. Obsolete view of performance. Need well-designed software for good speed already today.
The community convince itself that branch prediction is important for performance?

1980s insn sets, CPU costs → 1990s compilers, applications, data volumes, compiled code → 1990s/2000s hype (e.g., “Since programs typically encounter branches every 4–6 instructions, inaccurate branch prediction causes a severe performance degradation in highly superscalar or deeply pipelined designs”) → 2000s/2010s beliefs.

The fundamental question: Can a well-designed insn set with well-designed software remove the speed incentive for branch prediction?

“We need to look at current insn sets.” — Yes, interesting short-term question. Not my question in this talk.

“We need to look at badly written software.” — No. Obsolete view of performance. Need well-designed software for good speed already today.
How did the community convince itself that branch prediction is important for performance?

1980s insn sets, CPU costs $\rightarrow$ 1990s compilers, applications, data volumes, compiled code $\rightarrow$ 1990s/2000s hype (e.g., “Since programs typically encounter branches every 4–6 instructions, inaccurate branch prediction causes a severe performance degradation in highly superscalar or deeply pipelined designs”) $\rightarrow$ 2000s/2010s beliefs.

The fundamental question: Can a well-designed insn set with well-designed software remove the speed incentive for branch prediction?

“We need to look at current insn sets.” — Yes, interesting short-term question. Not my question in this talk.

“We need to look at badly written software.” — No. Obsolete view of performance. Need well-designed software for good speed already today.

“Fundamentally, you cannot compute branches in advance for these important computations. Look at, e.g., int32[n] heapsort. Inspect data, branch, repeat.”
The fundamental question: Can a well-designed insn set with well-designed software remove the speed incentive for branch prediction?

“We need to look at current insn sets.” — Yes, interesting short-term question. Not my question in this talk.

“We need to look at badly written software.” — No. Obsolete view of performance. Need well-designed software for good speed already today.

“Fundamentally, you cannot compute branches in advance for these important computations. Look at, e.g., int32[n] heapsort. Inspect data, branch, repeat.”
The fundamental question: Can a well-designed insn set with well-designed software remove the speed incentive for branch prediction?

“We need to look at current insn sets.” — Yes, interesting short-term question. Not my question in this talk.

“We need to look at badly written software.” — No. Obsolete view of performance. Need well-designed software for good speed already today.

“Fundamentally, you cannot compute branches in advance for these important computations. Look at, e.g., int32[n] heapsort. Inspect data, branch, repeat.”
The fundamental question:
Can a well-designed insn set with well-designed software remove the speed incentive for branch prediction?

“We need to look at current insn sets.” — Yes, interesting short-term question. Not my question in this talk.

“We need to look at badly written software.” — No. Obsolete view of performance. Need well-designed software for good speed already today.

“Fundamentally, you cannot compute branches in advance for these important computations. Look at, e.g., int32[n] heapsort. Inspect data, branch, repeat.”

— The current speed records for int32[n] sorting on Intel CPUs are held by sorting networks! Data-independent branches defined purely by n. Performance, parallelizability, predictability have clear connections.

sorting.cr.yp.to: software + verification tools.