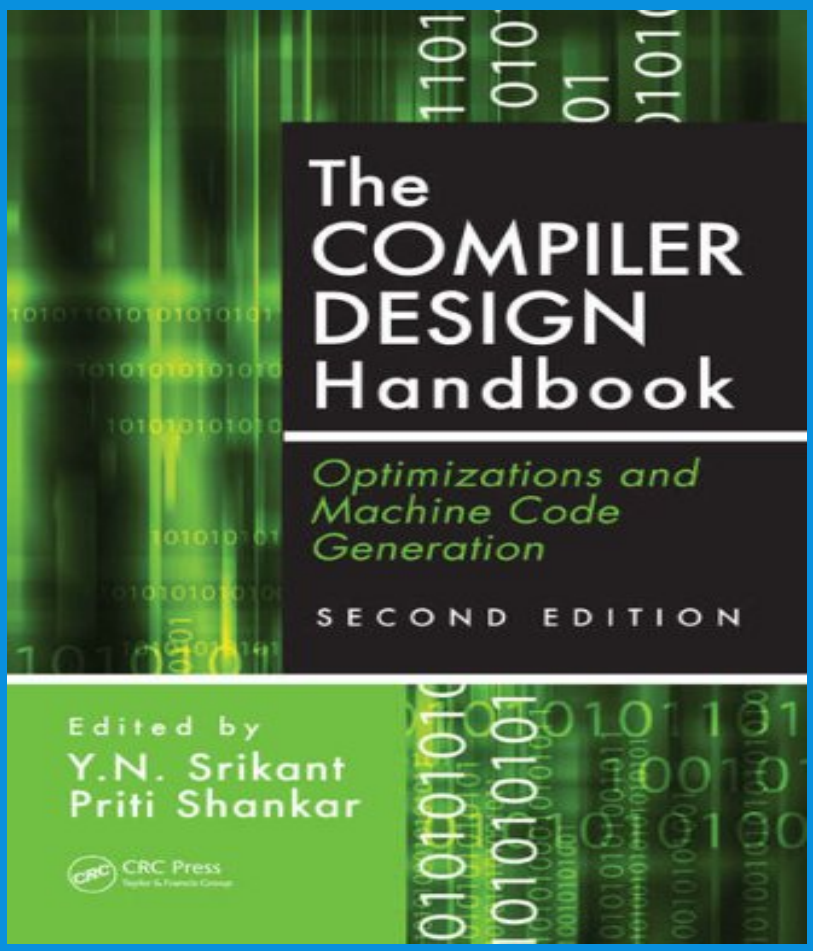
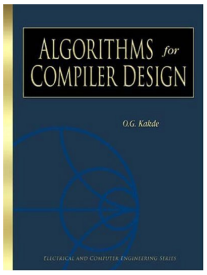
The compiler design handbook optimizations and machine code generation 2nd Edition Y.N. Srikant



Algorithms for compiler design 1st Edition O G Kakde



https://ebookname.com/product/algorithms-for-compiler-design-1st-edition-o-g-kakde/

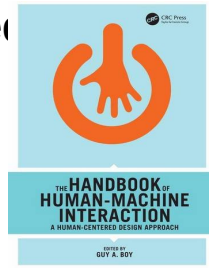
ebookname.com

Machine Design 5th Edition Alfred Hall

https://ebookname.com/product/machine-design-5th-edition-alfred-hall/

ebookname.com

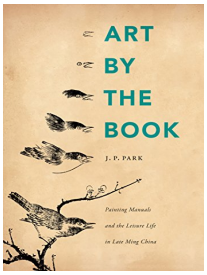
The Handbook of Human Machine Interaction A Human Centere(Design Approach 1st Edition Guy A. Boy (Editor)



https://ebookname.com/product/the-handbook-of-human-machine-interaction-a-human-centered-design-approach-1st-edition-guy-a-boy-editor/

ebookname.com

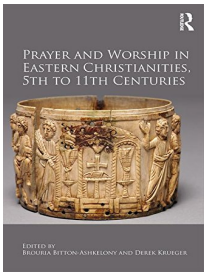
Art by the Book Painting Manuals and the Leisure Life in Late Ming China 1st Edition J.P. Park



https://ebookname.com/product/art-by-the-book-painting-manuals-and-the-leisure-life-in-late-ming-china-1st-edition-j-p-park/

ebookname.com

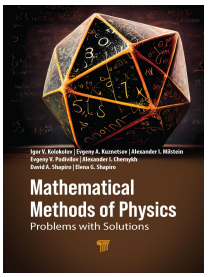
Prayer and Worship in Eastern Christianities 5th to 11th Centuries 1st Edition Brouria Bitton-Ashkelony



https://ebookname.com/product/prayer-and-worship-in-eastern-christianities-5th-to-11th-centuries-1st-edition-brouria-bitton-ashkelony/

ebookname.com

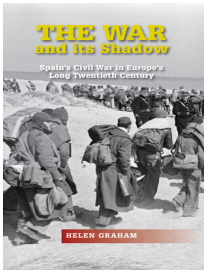
Mathematical Methods of Physics Problems with Solutions 1st Edition Igor V. Kolokolov & Evgeny A. Kuznetsov & Alexander I. Milstein & Evgeny V. Podivilov & Alexander I. Chernykh & David A. Shapiro & Elena G. Shapiro



https://ebookname.com/product/mathematical-methods-of-physics-problems-with-solutions-1st-edition-igor-v-kolokolov-evgeny-a-kuznetsov-alexander-i-milstein-evgeny-v-podivilov-alexander-i-chernykh-david-a-shapiro-e/

ebookname.com

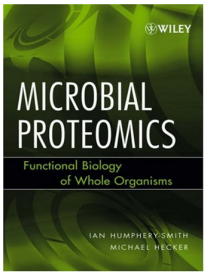
The War and Its Shadow Spain s Civil War in Europe s Long Twentieth Century Helen Graham



https://ebookname.com/product/the-war-and-its-shadow-spain-s-civil-war-in-europe-s-long-twentieth-century-helen-graham/

ebookname.com

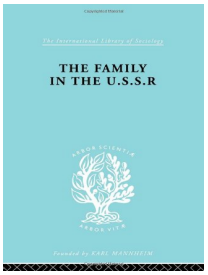
Microbial Proteomics Functional Biology of Whole Organisms 1st Edition Ian Humphery-Smith



https://ebookname.com/product/microbial-proteomics-functional-biology-of-whole-organisms-1st-edition-ian-humphery-smith/

ebookname.com

The Sociology of the Soviet Union The Family in the USSR Rudolf Schlesinger

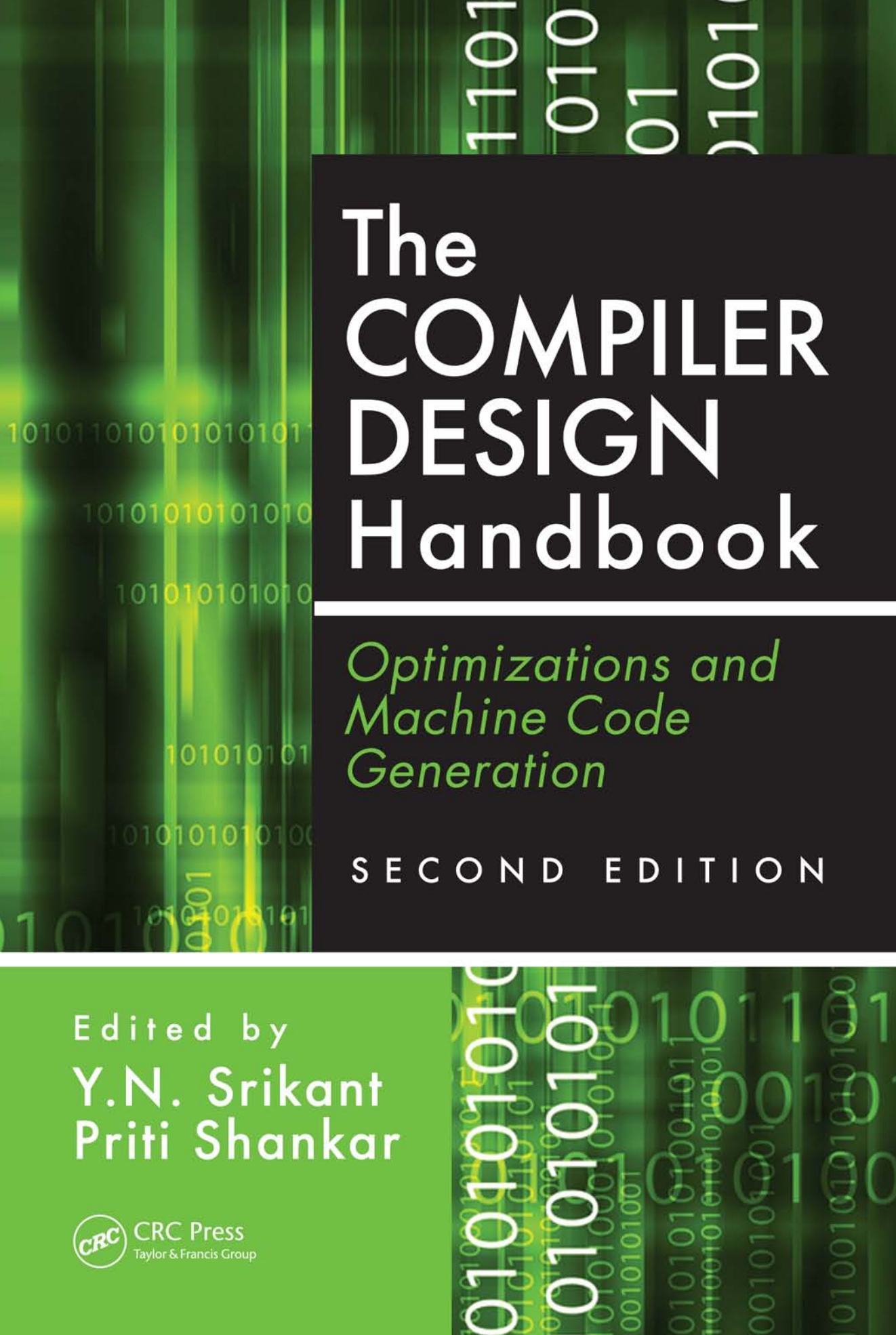


https://ebookname.com/product/the-sociology-of-the-soviet-union-the-family-in-the-ussr-rudolf-schlesinger/

ebookname.com

White Terror Jamie Bisher

https://ebookname.com/product/white-terror-jamie-bisher/



The COMPILER DESIGN Handbook

Optimizations and Machine Code Generation

SECOND EDITION

Edited by
Y.N. Srikant
Priti Shankar

CRC Press
Taylor & Francis Group

The COMPILER DESIGN Handbook

Optimizations and Machine Code Generation

SECOND EDITION

The COMPILER DESIGN Handbook

Optimizations and Machine Code Generation

SECOND EDITION

Edited by
Y.N. Srikant
Priti Shankar

CRC Press
Taylor & Francis Group
Boca Raton London New York

CRC Press is an imprint of the
Taylor & Francis Group, an **informa** business

CRC Press
Taylor & Francis Group
6000 Broken Sound Parkway NW, Suite 300
Boca Raton, FL 33487-2742

Printed in the United States of America on acid-free paper
10 9 8 7 6 5 4 3 2 1

International Standard Book Number-13: 978-1-4200-4382-2 (Hardcover)

Library of Congress Cataloging-in-Publication Data

The Compiler design handbook : optimizations and machine code generation / edited by Y.N. Srikant and Priti Shankar. -- 2nd ed.
p. cm.
Includes bibliographical references and index.
ISBN 978-1-4200-4382-2 (alk. paper)
1. Compilers (Computer programs) 2. Code generators. I. Srikant, Y. N. II. Shankar, P. (Priti) III. Title.

QA76.76.C65C35 2007
005.4'53--dc22 2007018733

Visit the Taylor & Francis Web site at
http://www.taylorandfrancis.com

and the CRC Press Web site at
http://www.crcpress.com

Table of Contents

1

Worst-Case Execution Time and Energy Analysis

Tulika Mitra
and
Abhik Roychoudhury
Department of Computer Science, School of Computing, National University of Singapore, Singapore
tulika@comp.nus.edu.sg and abhik@comp.nus.edu.sg

1.1 Introduction

Timing predictability is extremely important for hard real-time embedded systems employed in application domains such as automotive electronics and avionics. Schedulability analysis techniques can guarantee the satisfiability of timing constraints for systems consisting of multiple concurrent tasks. One of the key inputs required for the schedulability analysis is the worst-case execution time (WCET) of each of the tasks. WCET of a task on a target processor is defined as its maximum execution time across all possible inputs.

Figure 1.1a and Figure 1.2a show the variation in execution time of a `quick sort` program on a simple and complex processor, respectively. The program sorts a five-element array. The figures show the distribution of execution time (in processor cycles) for all possible permutations of the array elements as inputs. The maximum execution time across all the inputs is the WCET of the program. This simple example illustrates the inherent difficulty of finding the WCET value:

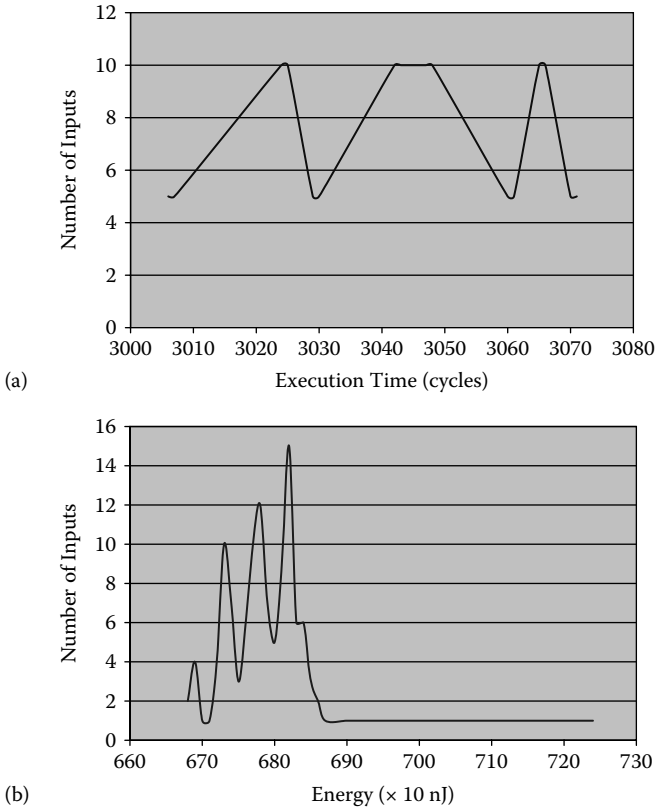


FIGURE 1.1 Distribution of time and energy for different inputs of an application on a simple processor.

- Clearly, executing the program for all possible inputs so as to bound its WCET is not feasible. The problem would be trivial if the worst-case input of a program is known *a priori*. Unfortunately, for most programs the worst-case input is unknown and cannot be derived easily.
- Second, the complexity of current micro-architectures implies that the WCET is heavily influenced by the target processor. This is evident from comparing Figure 1.1a with Figure 1.2a. Therefore, the timing effects of micro-architectural components have to be accurately accounted for.

Static analysis methods estimate a bound on the WCET. These analysis techniques are conservative in nature. That is, when in doubt, the analysis assumes the worst-case behavior to guarantee the safety of the estimated value. This may lead to overestimation in some cases. Thus, the goal of static analysis methods is to estimate a *safe* and *tight* WCET value. Figure 1.3 explains the notion of safety and tightness in the context of static WCET analysis. The figure shows the variation in execution time of a task. The *actual WCET* is the maximum possible execution time of the program. The static analysis method generates the *estimated WCET* value such that *estimated WCET* $\geq$ *actual WCET*. The difference between the estimated and the actual WCET is the overestimation and determines how tight the estimation is. Note that the static analysis methods guarantee that the estimated WCET value can never be less than the actual WCET value. Of course, for a complex task running on a complex processor, the actual WCET value is unknown. Instead, simulation or execution of the program with a subset of possible inputs generates the *observed WCET*, where *observed WCET* $\leq$ *actual WCET*. In other words, the observed WCET value is not safe, in the sense that it cannot be used to provide absolute timing guarantees for safety-critical systems. A notion related to WCET is the *BCET (best-case execution time)*, which represents the minimum execution time across all possible inputs. In this chapter, we will focus on static analysis

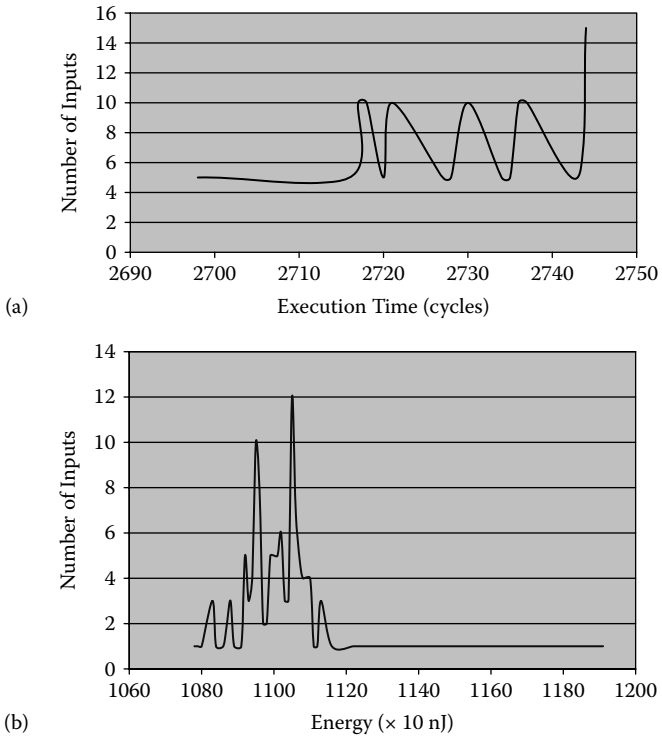


FIGURE 1.2 Distribution of time and energy for different inputs of the same application on a complex processor.

techniques to estimate the WCET. However, the same analysis methods can be easily extended to estimate the BCET.

Apart from timing, the proliferation of battery-operated embedded devices has made energy consumption one of the key design constraints. Increasingly, mobile devices are demanding improved functionality and higher performance. Unfortunately, the evolution of battery technology has not been able to keep up with performance requirements. Therefore, designers of mission-critical systems, operating on limited battery life, have to ensure that both the timing and the energy constraints are satisfied under all possible

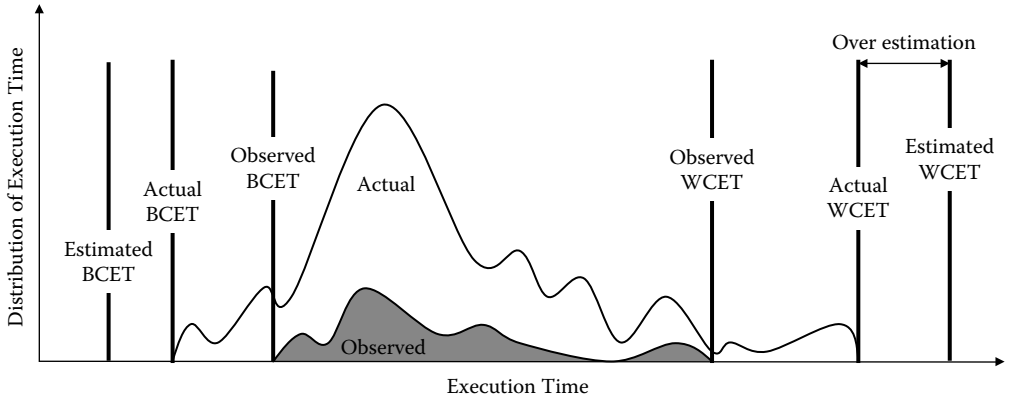


FIGURE 1.3 Definition of observed, actual, and estimated WCET.

scenarios. The battery should never drain out before a task completes its execution. This concern leads to the related problem of estimating the worst-case energy consumption of a task running on a processor for all possible inputs. Unlike WCET, estimating the worst-case energy remains largely unexplored even though it is considered highly important [86], especially for mobile devices. Figure 1.1b and Figure 1.2b show the variation in energy consumption of the `quick sort` program on a simple and complex processor, respectively.

A natural question that may arise is the possibility of using the WCET path to compute a bound on the worst-case energy consumption. As *energy* = *average power* × *execution time*, this may seem like a viable solution and one that can exploit the extensive research in WCET analysis in a direct fashion. Unfortunately, the path corresponding to the WCET may not coincide with the path consuming maximum energy. This is made apparent by comparing the distribution of execution time and energy for the same program and processor pair as shown in Figure 1.1 and Figure 1.2. There are a large number of input pairs $\langle I_1, I_2 \rangle$ in this program, where $time(I_1) < time(I_2)$, but $energy(I_1) > energy(I_2)$. This happens as the energy consumed because of the switching activity in the circuit need not necessarily have a correlation with the execution time. Thus, the input that leads to WCET may not be identical to the input that leads to the worst-case energy.

The execution time or energy is affected by the path taken through the program and the underlying micro-architecture. Consequently, static analysis for worst-case execution time or energy typically consists of three phases. The first phase is the *program path analysis* to identify loop bounds and infeasible flows through the program. The second phase is the *architectural modeling* to determine the effect of pipeline, cache, branch prediction, and other components on the execution time (energy). The last phase, *estimation*, finds an upper bound on the execution time (energy) of the program given the results of the flow analysis and the architectural modeling.

Recently, there has been some work on *measurement-based timing analysis* [6, 17, 92]. This line of work is mainly targeted toward soft real-time systems, such as multimedia applications, that can afford to miss the deadline once in a while. In other words, these application domains do not require absolute timing guarantees. Measurement-based timing analysis methods execute or simulate the program on the target processor for a subset of all possible inputs. They derive the maximum observed execution time (see the definition in Figure 1.3) or the distribution of execution time from these measurements. Measurement-based performance analysis is quite useful for soft real-time applications, but they may underestimate the WCET, which is not acceptable in the context of safety-critical, hard real-time applications. In this article, we only focus on static analysis techniques that provide safe bounds on WCET and worst-case energy. The analysis methods assume uninterrupted program execution on a single processor. Furthermore, the program being analyzed should be free from unbounded loops, unbounded recursion, and dynamic function calls [67].

The rest of the chapter is organized as follows. We proceed with programming-language-level WCET analysis in the next section. This is followed by micro-architectural modeling in Section 1.3. We present a static analysis technique to estimate worst-case energy bound in Section 1.4. A brief description of existing WCET analysis tools appears in Section 1.5, followed by conclusions.

1.2 Programming-Language-Level WCET Analysis

We now proceed to discuss static analysis methods for estimating the WCET of a program. For WCET analysis of a program, the first issue that needs to be determined is the program representation on which the analysis will work. Earlier works [73] have used the *syntax tree* where the (nonleaf) nodes correspond to programming-language-level control structures. The leaves correspond to basic blocks — maximal fragments of code that do not involve any control transfer. Subsequently, almost all work on WCET analysis has used the *control flow graph*. The nodes of a control flow graph (CFG) correspond to basic blocks, and the edges correspond to control transfer between basic blocks. When we construct the CFG of a program, a separate copy of the CFG of a function f is created for every distinct call site of f in the program

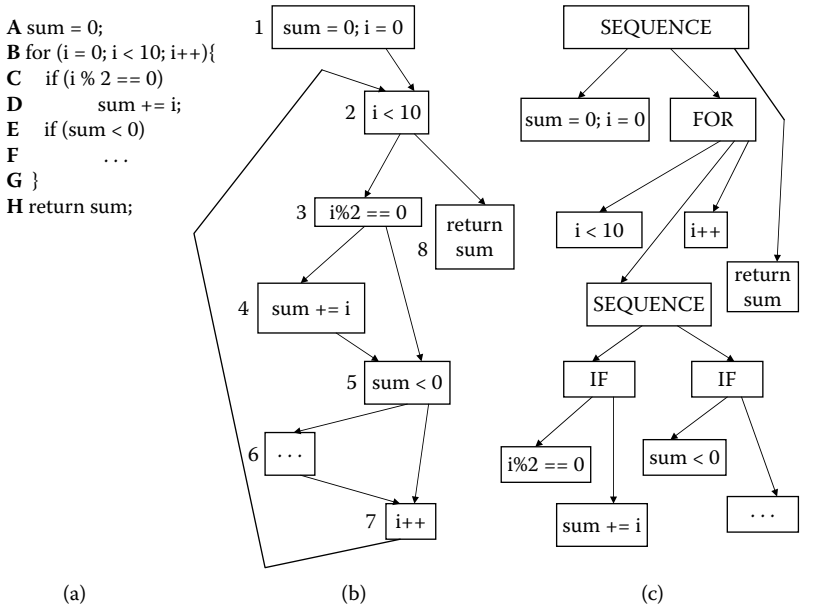


FIGURE 1.4 (a) A code fragment. (b) Control flow graph of the code fragment. (c) Syntax tree of the code fragment.

such that each call transfers control to its corresponding copy of CFG. This is how interprocedural analysis will be handled. Figure 1.4 shows a small code fragment as well as its syntax tree and control flow graph representations.

One important issue needs to be clarified in this regard. The control flow graph of a program can be either at the source code level or at the assembly code level. The difference between the two comes from the compiler optimizations. Our program-level analysis needs to be hooked up with micro-architectural modeling, which accurately estimates the execution time of each instruction while considering the timing effects of underlying microarchitectural features. Hence we always consider the assembly-code-level CFG. However, while showing our examples, we will show CFG at the source code level for ease of exposition.

1.2.1 WCET Calculation

We explain WCET analysis methods in a top-down fashion. Consequently, at the very beginning, we present *WCET calculation* — how to combine the execution time estimates of program fragments to get the execution time estimate of a program. We assume that the *loop bounds* (i.e., the maximum number of iterations for a loop) are known for every program loop; in Section 1.2.2 we outline some methods to estimate loop bounds.

In the following, we outline the three main categories of WCET calculation methods. The path-based and integer linear programming methods operate on the program's control flow graph, while the tree-based methods operate on the program's syntax tree.

1.2.1.1 Tree-Based Methods

One of the earliest works on software timing analysis was the work on *timing schema* [73]. The technique proceeds essentially by a bottom-up pass of the syntax tree. During the traversal, it associates an execution time estimate for each node of the tree. The execution time estimate for a node is obtained from the execution time estimates of its children, by applying the rules in the schema. The schema prescribes

rules — one for each control structure of the programming language. Thus, rules corresponding to a sequence of statements, *if-then-else* and *while-loop* constructs, can be described as follows.

- *Time*(`S1; S2`) = *Time*(`S1`) + *Time*(`S2`)
- *Time*(`if (B) { S1 } else { S2}`) = *Time*(`B`) + max(*Time*(`S1`), *Time*(`S2`))
- *Time* (`while (B) { S1}`) = $(n+1)$**Time*(`B`) + n**Time*(`S1`)

Here, n is the loop bound. Clearly, `S1`, `S2` can be complicated code fragments whose execution time estimates need to obtained by applying the schema rules for the control structures appearing in `S1`, `S2`. Extensions of the timing schema approach to consider micro-architectural modeling will be discussed in Section 1.3.5.

The biggest advantage of the timing schema approach is its simplicity. It provides an efficient compositional method for estimating the WCET of a program by combining the WCET of its constituent code fragments. Let us consider the following schematic code fragment Pgm. For simplicity of exposition, we will assume that all assignments and condition evaluations take one time unit.

```
i = 0; while (i<100) {if (B') S1 else S2; i++;}
```

If *Time*(`S1`) > *Time*(`S2`), by using the rule for if-then-else statements in the timing schema we get

$$\textit{Time}(\texttt{if (B') S1 else S2 }) = \textit{Time}(\texttt{B'}) + \textit{Time}(\texttt{S1}) = 1 + \textit{Time}(\texttt{S1})$$

Now, applying the rule for while-loops in the timing schema, we get the following. The loop bound in this case is 100.

$$\begin{aligned}\textit{Time}(\texttt{while (i<100) \{if(B') S1 else S2\}}) &= 101 * \textit{Time}(\texttt{i} < 100) + \\ &\quad 100 * \textit{Time}(\texttt{if (B') S1 else S2}) \\ &= 101 * 1 + 100 * (1 + \textit{Time}(\texttt{S1})) \\ &= 201 + 100 * \textit{Time}(\texttt{S1})\end{aligned}$$

Finally, using the rule for sequential composition in the timing schema we get

$$\begin{aligned}\textit{Time}(Pgm) &= \textit{Time}(\texttt{i = 0}) + \textit{Time}(\texttt{while (i<100) \{if (B') S1 else S2\}}) \\ &= 1 + 201 + 100 * \textit{Time}(\texttt{S1}) = 202 + 100 * \textit{Time}(\texttt{S1})\end{aligned}$$

The above derivation shows the working of the timing schema. It also exposes one of its major weaknesses. In the timing schema, the timing rules for a program statement are local to the statement; they do not consider the *context* with which the statement is arrived at. Thus, in the preceding we estimated the maximum execution time of `if (B') S1 else S2` by taking the execution time for evaluating `B` and the time for executing `S1` (since time for executing `S1` is greater than the time for executing `S2`). As a result, since the if-then-else statement was inside a loop, our maximum execution time estimate for the loop considered the situation where `S1` is executed in every loop iteration (i.e., the condition `B'` is evaluated to true in every loop iteration).

However, in reality `S1` may be executed in very few loop iterations for any input; if *Time*(`S1`) is significantly greater than *Time*(`S2`), the result returned by timing schema will be a gross overestimate. More importantly, it is difficult to extend or augment the timing schema approach so that it can return tighter estimates in such situations. In other words, even if the user can provide the information that "it is infeasible to execute `S1` in every loop iteration of the preceding program fragment Pgm," it is difficult to exploit such information in the timing schema approach. Difficulty in exploiting infeasible program flows information (for returning tighter WCET estimates) remains one of the major weaknesses of the timing schema. We will revisit this issue in Section 1.2.2.

1.2.1.2 Path-Based Methods

The path-based methods perform WCET calculation of a program P via a longest-path search over the control flow graph of P. The loop bounds are used to prevent unbounded unrolling of the loops. The

biggest disadvantage of this method is its complexity, as in the worst-case it may amount to enumeration of all program paths that respect the loop bounds. The advantage comes from its ability to handle various kinds of flow information; hence, infeasible path information can be easily integrated with path-based WCET calculation methods.

One approach for restricting the complexity of longest-path searches is to perform symbolic state exploration (as opposed to an explicit path search). Indeed, it is possible to cast the path-based searches for WCET calculation as a (symbolic) model checking problem [56]. However, because model checking is a verification method [13], it requires a temporal property to verify. Thus, to solve WCET analysis using model-checking-based verification, one needs to guess possible WCET estimates and verify that these estimates are indeed WCET estimates. This makes model-checking-based approaches difficult to use (see [94] for more discussion on this topic). The work of Schuele and Schneider [72] employs a *symbolic* exploration of the program's underlying transition system for finding the longest path, without resorting to checking of a temporal property. Moreover, they [72] observe that for finding the WCET there is no need to (even symbolically) maintain data variables that do not affect the program's control flow; these variables are identified via program slicing. This leads to overall complexity reduction of the longest-path search involved in WCET calculation.

A popular path-based WCET calculation approach is to employ an explicit longest-path search, but over a fragment of the control flow graph [31, 76, 79]. Many of these approaches operate on an acyclic fragment of the control flow graph. Path enumeration (often via a breadth-first search) is employed to find the longest path within the acyclic fragment. This could be achieved by a weighted longest-path algorithm (the weights being the execution times of the basic blocks) to find the longest sequence of basic blocks in the control flow graph for a program fragment. The longest-path algorithm can be obtained by a variation of Djikstra's shortest-path algorithm [76]. The longest paths obtained in acyclic control flow graph fragments are then combined with the loop bounds to yield the program's WCET. The path-based approaches can readily exploit any known infeasible flow information. In these methods, the explicit path search is pruned whenever a known infeasible path pattern is encountered.

1.2.1.3 Integer Linear Programming (ILP)

ILP combines the advantages of the tree and path-based approaches. It allows (limited) integration of infeasible path information while (often) being much less expensive than the path-based approaches. Many existing WCET tools such as aiT [1] and Chronos [44] employ ILP for WCET calculation.

The ILP approach operates on the program's control flow graph. Each basic block B in the control flow graph is associated with an integer variable N_B, denoting the *total* execution count of basic block B. The program's WCET is then given by the (linear) objective function

$$\text{maximize} \sum_{B \in \mathcal{B}} N_B * c_B$$

where $\mathcal{B}$ is the set of basic blocks of the program, and c_B is a constant denoting the WCET estimate of basic block B. The linear constraints on N_B are developed from the flow equations based on the control flow graph. Thus, for basic block B,

$$\sum_{B' \to B} E_{B' \to B} = N_B = \sum_{B \to B''} E_{B \to B''}$$

where $E_{B' \to B}$ ($E_{B \to B''}$) is an ILP variable denoting the number of times control flows through the control flow graph edge $B' \to B$ ($B \to B''$). Additional linear constraints are also provided to capture loop bounds and any known infeasible path information.

In the example of Figure 1.4, the control flow equations are given as follows. We use the numbering of the basic blocks 1 to 8 shown in Figure 1.4. Let us examine a few of the control flow equations. For basic block 1, there are no incoming edges, but there is only one outgoing edge $1 \to 2$. This accounts for the constraint $N_1 = E_{1 \to 2}$; that is, the number of executions of basic block 1 is equal to the number of flows

from basic block 1 to basic block 2. In other words, whenever basic block 1 is executed, control flows from basic block 1 to basic block 2. Furthermore, since basic block 1 is the entry node, it is executed exactly once; this is captured by the constraint $N_1 = 1$. Now, let us look at the constraints for basic block 2; the inflows to this basic block are the edges $1 \rightarrow 2$ and $7 \rightarrow 2$ and the outflows are the edges $2 \rightarrow 3$ and $2 \rightarrow 8$. This means that whenever block 2 is executed, control must have flown in via either the edge $1 \rightarrow 2$ or the edge $7 \rightarrow 2$; this accounts for the constraint $E_{1\rightarrow 2} + E_{7\rightarrow 2} = N_2$. Furthermore, whenever block 2 is executed, control must flow out via the edge $2 \rightarrow 3$ or the edge $2 \rightarrow 8$. This accounts for the constraint $N_2 = E_{2\rightarrow 3} + E_{2\rightarrow 8}$. The inflow/outflow constraints for the other basic blocks are obtained in a similar fashion. The full set of inflow/outflow constraints for Figure 1.4 are shown in the following.

$$
\begin{aligned}
N_1 = 1 &= E_{1\rightarrow 2} \\
E_{1\rightarrow 2} + E_{7\rightarrow 2} = N_2 &= E_{2\rightarrow 3} + E_{2\rightarrow 8} \\
E_{2\rightarrow 3} = N_3 &= E_{3\rightarrow 4} + E_{3\rightarrow 5} \\
E_{3\rightarrow 4} = N_4 &= E_{4\rightarrow 5} \\
E_{4\rightarrow 5} + E_{3\rightarrow 5} = N_5 &= E_{5\rightarrow 6} + E_{5\rightarrow 7} \\
E_{5\rightarrow 6} = N_6 &= E_{6\rightarrow 7} \\
E_{6\rightarrow 7} + E_{5\rightarrow 7} = N_7 &= E_{7\rightarrow 2} \\
E_{2\rightarrow 8} = N_8 &= 1
\end{aligned}
$$

The execution time of the program is given by the following linear function in N_i variables (c_i is a *constant* denoting the WCET of basic block i).

$$\sum_{i=1}^{8} N_i * c_i$$

Now, if we ask the ILP solver to maximize this objective function subject to the inflow/outflow constraints, it will not succeed in producing a time bound for the program. This is because the only loop in the program has not been bounded. The *loop bound* information itself must be provided as linear constraints. In this case, since Figure 1.4 has only one loop, this accounts for the constraint

$$E_{7\rightarrow 2} \leq 10$$

Using this loop bound, the ILP solver can produce a WCET bound for the program. Of course, the WCET bound can be tightened by providing additional linear constraints capturing infeasible path information; the flow constraints by default assume that all paths in the control flow graph are feasible. It is worthwhile to note that the ILP solver is capable of only *utilizing* the loop bound information and other infeasible path information that is provided to it as linear constraints. Inferring the loop bounds and various infeasible path patterns is a completely different problem that we will discuss next.

Before moving on to infeasible path detection, we note that tight execution time estimates for basic blocks (the constants c_i appearing in the ILP objective function) are obtained by micro-architectural modeling techniques described in Section 1.3. Indeed, this is how the micro-architectural modeling and program path analysis hook up in most existing WCET estimation tools. The program path analysis is done by an ILP solver; infeasible path and loop bound information are integrated with the help of additional linear constraints. The objective function of the ILP contains the WCET estimates of basic blocks as constants. These estimates are provided by micro-architectural modeling, which considers cache, pipeline, and branch prediction behavior to tightly estimate the maximum possible execution time of a basic block B (where B is executed in any possible hardware state and/or control flow context).

1.2.2 Infeasible Path Detection and Exploitation

In the preceding, we have described WCET calculation methods without considering that certain sequences of program fragments may be *infeasible*, that is, not executed on any program input. Our WCET calculation methods only considered the loop bounds to determine a program's WCET estimate. In reality, the WCET calculation needs to consider (and exploit) other information about infeasible program paths. Moreover, the loop bounds also need to be estimated through an off-line analysis. Before proceeding further, we define the notion of an infeasible path.

Definition 1.1
Given a program P, let $\mathcal{B}_P$ be the set of basic blocks of P. Then, an infeasible path of P is a sequence of basic blocks σ over the alphabet $\mathcal{B}_P$, such that σ does not appear in the execution trace corresponding to any input of P.

Clearly, knowledge of infeasible path patterns can tighten WCET estimates. This is simply because the longest path determined by our favorite WCET calculation method may be an infeasible one. Our goal is to efficiently detect *and* exploit infeasible path information for WCET analysis. The general problem of infeasible path detection is NP-complete [2]. Consequently, any approach toward infeasible path detection is an underapproximation — any path determined to be infeasible is indeed infeasible, but not vice versa.

It is important to note that the infeasible path information is often given at the level of source code, whereas the WCET calculation is often performed at the assembly-code-level control flow graph. Because of compiler optimizations, the control flow graph at the assembly code level is not the same as the control flow graph at the source code level. Consequently, infeasible path information that is (automatically) inferred or provided (by the user) at the source code level needs to be converted to a lower level within a WCET estimation tool. This transformation of flow information can be automated and integrated with the compilation process, as demonstrated in [40].

In the following, we discuss methods for infeasible path *detection*. Exploitation of infeasible path information will involve augmenting the WCET calculation methods we discussed earlier. At this stage, it is important to note that infeasible path detection typically involves a smart path search in the program's control flow graph. Therefore, if our WCET calculation proceeds by path-based methods, it is difficult to separate the infeasible path detection and exploitation. In fact, for many path-based methods, the WCET detection and exploitation will be fused into a single step. Consequently, we discuss infeasible path detection methods and along with it exploitation of these in path-based WCET calculation. Later on, we also discuss how the other two WCET calculation approaches (tree-based methods and ILP-based methods) can be augmented to exploit infeasible path information. We note here that the problem of infeasible path detection is a very general one and has implications outside WCET analysis. In the following, we only capture some works as representatives of the different approaches to solving the problem of infeasible path detection.

1.2.2.1 Data Flow Analysis

One of the most common approaches for infeasible path detection is by adapting data flow analysis [21, 27]. In this analysis, each control location in the program is associated with an *environment*. An environment is a mapping of program variables to values, where each program variable is mapped to a *set of values*, instead of a single value. The environment of a control location L captures *all* the possible values that the program variables may assume at L; it captures variable valuations for all possible visits to L. Thus, if x is an integer variable, and at line 70 of the program, the environment at line 70 maps x to $[0.5]$, this means that x is guaranteed to assume an integer value between 0 and 5 when line 70 is visited. An infeasible path is detected when a variable is mapped to the empty set of values at a control location.

Approaches based on data flow analysis are often useful for finding a wide variety of infeasible paths and loop bounds. However, the environments computed at a control location may be too approximate. It is important to note that the environment computed at a control location CL is essentially an *invariant*

property — a property that holds for *every* visit to CL. To explain this point, consider the example program in Figure 1.4a. Data flow analysis methods will infer that in line E of the program `sum` $\in [0..20]$, that is, $0 \leq$ `sum` ≤ 20. Hence we can infer that execution of lines E, F in Figure 1.4a constitutes an infeasible path. However, by simply keeping track of all possible variable values at each control location we cannot directly infer that line D of Figure 1.4a cannot be executed in consecutive iterations of the loop.

1.2.2.2 Constraint Propagation Methods

The above problem is caused by the merger of environments at any control flow merge point in the control flow graph. The search in data flow analysis is not truly path sensitive — at any control location CL we construct the environment for CL from the environments of all the control locations from which there is an incoming control flow to CL. One way to solve this problem is to perform constraint propagation [7, 71] (or value propagation as in [53]) along paths via *symbolic* execution. Here, instead of assigning possible values to program variables (as in flow analysis), each input variable is given a special value: *unknown*. Thus, if nothing is known about a variable `x`, we simply represent it as `x`. The operations on program variables will then have to deal with these symbolic representations of variables. The search then accumulates constraints on `x` and detects infeasible paths whenever the constraint store becomes unsatisfiable. In the program of Figure 1.4a, by traversing lines C,D we accumulate the constraint `i % 2` $\neq$ `0`. In the subsequent iteration, we accumulate the constraint `i+1 % 2` $\neq$ `0`. Note that via symbolic execution we know that the current value of `i` is one greater than the value in the previous iteration, so the constraint `i+1 % 2` $\neq$ `0`. We now need to show that the constraint `i % 2` $\neq$ `0` $\wedge$ `i+1 % 2` $\neq$ `0` is unsatisfiable in order to show that line D in Figure 1.4a cannot be visited in subsequent loop iterations. This will require the help of external constraint solvers or theorem provers such as Simplify [74]. Whether the constraint in question can be solved automatically by the external prover, of course, depends on the prover having appropriate decision procedures to reason about the operators appearing in the constraint (such as the addition [+] and remainder [%] operators appearing in the constraint `i % 2` $\neq$ `0` $\wedge$ `i+1 % 2` $\neq$ `0`).

The preceding example shows the plus and minus points of using path-sensitive searches for infeasible path detection. The advantage of using such searches is the precision with which we can detect infeasible program paths. The difficulty in using full-fledged path-sensitive searches (such as model checking) is, of course, the huge number of program paths to consider.[1]

In summary, even though path-sensitive searches are more accurate, they suffer from a huge complexity. Indeed, this has been acknowledged in [53], which accommodates specific heuristics to perform path merging. Consequently, using path-sensitive searches for infeasible path detection does not scale up to large programs. Data flow analysis methods fare better in this regard since they perform merging at control flow merge points in the control flow graph. However, even data flow analysis methods can lead to full-fledged loop unrolling if a variable gets new values in every iteration of a loop (e.g., consider the program `while (...){ i++ }`).

1.2.2.3 Heuristic Methods

To avoid the cost of loop unrolling, the WCET community has studied techniques that operate on the acyclic graphs representing the control flow of a single loop iteration [31, 76, 79]. These techniques do not detect or exploit infeasible paths that span across multiple loop iterations. The basic idea is to find the weighted longest path in any loop iteration and multiply its cost with the loop bound. Again, the complication arises from the presence of infeasible paths even within a loop iteration. The work of Stappert et al. [76] finds the longest path π in a loop iteration and checks whether it is feasible; if π is infeasible, it employs

[1] Furthermore, the data variables of a program typically come from unbounded domains such as integers. Thus, use of a finite-state search method such as model checking will have to either employ data abstractions to construct a finite-state transition system corresponding to a program or work on symbolic state representations representing infinite domains (possibly as constraints), thereby risking nontermination of the search.

graph-theoretic methods to remove π from the control flow graph of the loop. The longest-path calculation is then run again on the modified graph. This process is repeated until a feasible longest path is found. Clearly, this method can be expensive if the feasible paths in a loop have relatively low execution times.

To address this gap, the recent work of Suhendra et al. [79] has proposed a more "infeasible path aware" search of the control flow graph corresponding to a loop body. In this work, the infeasible path detection and exploitation proceeds in two separate steps. In the first step, the work computes "conflict pairs," that is, incompatible (branch, branch) or (assignment, branch) pairs. For example, let us consider the following code fragment, possibly representing the body of a loop.

```
1 if (x > 3)
2     z = z + 1;
3 else
4     x = 1;
5 if (x < 2)
6     z = z/2;
7 else
8     z = z -1;
```

Clearly, the assignment at line 4 conflicts with the branch at line 5 evaluating to false. Similarly, the branch at line 1 evaluating to true conflicts with the branch at line 5 evaluating to true. Such conflicting pairs are detected in a traversal of the control flow directed acyclic graph (DAG) corresponding to the loop body. Subsequently, we traverse the control flow DAG of the loop body from sink to source, always keeping track of the heaviest path. However, if any assignment or branch decision appearing in the heaviest path is involved in a conflict pair, we also keep track of the next heaviest path that is not involved in such a pair. Consequently, we may need to keep track of more than one path at certain points during the traversal; however, redundant tracked paths are removed as soon as conflicts (as defined in the conflict pairs) are resolved during the traversal. This produces a path-based WCET calculation method that detects and exploits infeasible path patterns and still avoids expensive path enumeration or backtracking.

We note that to scale up infeasible path detection and exploitation to large programs, the notion of pairwise conflicts is important. Clearly, this will not allow us to detect that the following is an infeasible path:

```
x = 1; y = x; if (y > 2){...
```

However, using pairwise conflicts allows us to avoid full-fledged data flow analysis in WCET calculation. The work of Healy and Whalley [31] was the first to use pairwise conflicts for infeasible path detection and exploitation. Apart from pairwise conflicts, this work also detects iteration-based constraints, that is, the behavior of individual branches across loop iterations. Thus, if we have the following program fragment, the technique of Healy and Whalley [31] will infer that the branch inside the loop is true only for the iterations 0..24.

```
for (i = 0; i < 100; i++){
    if (i < 25)
                { S1; }
    else        { S2;}
}
```

If the time taken to execute `S1` is larger than the time taken to execute `S2`, we can estimate the cost of the loop to be $25 * Time(S1) + 75 * Time(S2)$. Note that in the absence of a framework for using iteration-based constraints, we would have returned the cost of the loop as $100 * Time(S1)$.

In principle, it is possible to combine the efficient control flow graph traversal in [79] with the framework in [31], which combines branch constraints as well as iteration-based constraints. This can result in a path-based WCET calculation that performs powerful infeasible path detection [31] and efficient infeasible path exploitation [79].

```
for (i = 1; i <= 100; i++){
        for (j = i; j <= 100; j++){
             ...
```

FIGURE 1.5 A nonrectangular loop nest.

1.2.2.4 Loop Bound Inferencing

An important part of infeasible path detection and exploitation is inferencing and usage of loop bounds. Without sophisticated inference of loop bounds, the WCET estimates can be vastly inflated. To see this point, we only need to examine a nested loop of the form shown in Figure 1.5. Here, a naive method will put the loop bound of the inner loop as $100 * 100 = 10,000$, which is a gross overestimate over the actual bound of $1 + 2 + \ldots + 100 = 5050$.

Initial work on loop bounds relied on the programmer to provide manual annotations [61]. These annotations are then used in the WCET calculation. However, giving loop bound annotations is in general an error-prone process. Subsequent work has integrated automated loop bound inferencing as part of infeasible path detection [21]. The work of Liu and Gomez [52] exploits the program structure for high-level languages (such as functional languages) to infer loop bounds. In this work, from the recursive structure of the functions in a functional program, a cost function is constructed automatically. Solving this cost-bound function can then yield bounds on loop executions (often modeled as recursion in functional programs). However, if the program is recursive (as is common for functional programs), the cost bound function is also recursive and does not yield a closed-form solution straightaway. Consequently, this technique [52] (a) performs symbolic evaluation of the cost-bound function using knowledge of program inputs and then (b) transforms the symbolically evaluated function to simplify its recursive structure. This produces the program's loop bounds. The technique is implemented for a subset of the functional language Scheme.[2]

For imperative programs, the work of Healy et al. [30] presents a comprehensive study for inferring loop bounds of various kinds of loops. It handles loops with multiple exits by automatically identifying the conditional branches within a loop body that may affect the number of loop iterations. Subsequently, for each of these branches the range of loop iterations where they can appear is detected; this information is used to compute the loop bounds. Moreover, the work of Healy et al. [30] also presents techniques for automatically inferring bounds on loops where loop exit/entry conditions depend on values of program variables. As an example, let us consider the nonrectangular loop nest shown in Figure 1.5. The technique of Healy et al. [30] will automatically extract the following expression for the bound on the number of executions of the inner loop.

$$N_{inner} = \sum_{i=1}^{100} \sum_{j=i}^{100} 1 = \sum_{i=1}^{100} \left(\sum_{j=1}^{100} 1 - \sum_{j=1}^{i-1} 1 \right) = \sum_{i=1}^{100} (100 - (i-1))$$

We can then employ techniques for solving summations to obtain N_{inner}.

1.2.2.5 Exploiting Infeasible Path Information in Tree-Based WCET Calculation

So far, we have outlined various methods for detecting infeasible paths in a program's control flow graph. These methods work by traversing the control flow graph and are closer to the path-based methods.

[2] Dealing loops as recursive procedures has also been studied in [55] but in a completely different context. This work uses context-sensitive interprocedural analysis to separate out the cache behavior of different executions of the recursive procedure corresponding to a loop, thereby distinguishing, for instance, the cache behavior of the first loop iteration from the remaining loop iterations.

If the WCET calculation is performed by other methods (tree based or ILP), how do we even integrate the infeasible path information into the calculation? In other words, if infeasible path patterns have been detected, how do we let tree-based or ILP-based WCET calculation exploit these patterns to obtain tighter WCET bounds? We first discuss this issue for tree-based methods and then for ILP methods.

One simple way to exploit infeasible path information is to partition the set of program inputs. For each input partition, the program is partially evaluated to remove the statements that are never executed (for inputs in that partition). Timing schema is applied to this partially evaluated program to get its WCET. This process is repeated for every input partition, thereby yielding a WCET estimate for each input partition. The program's WCET is set to the maximum of the WCETs for all the input partitions. To see the benefit of this approach, consider the following schematic program with a boolean input `b`.

```
IfStmt1 :  if (b == 0) { S1;} else {S2;}
IfStmt2 :  if (b == 1) { S3;} else {S4;}
```

Assume that

$$Time(\texttt{S1}) > Time(\texttt{S2}) \text{ and } Time(\texttt{S3}) > Time(\texttt{S4})$$

Then using the rules of timing schema we have the following. For convenience, we call the first (second) if statement in the preceding schematic program fragment $IfStmt_1$ ($IfStmt_2$).

$$\begin{aligned}
&Time(IfStmt_1) = && Time(\texttt{b == 0}) + Time(\texttt{S1})\\
&Time(IfStmt_2) = && Time(\texttt{b == 1}) + Time(\texttt{S3})\\
&Time(IfStmt_1; IfStmt_2) = && Time(IfStmt_1) + Time(IfStmt_2) =\\
& && Time(\texttt{b == 0}) + Time(\texttt{b == 1}) + Time(\texttt{S1}) + Time(\texttt{S3})
\end{aligned}$$

We now consider the execution time for the two possible inputs and take their maximum. Let us now consider the program for input `b` = 0. Since statements `S1` and `S4` are executed, we have:

$$Time(IfStmt_1; IfStmt_2)_{\texttt{b=0}} = Time(\texttt{b == 0}) + Time(\texttt{b == 1}) + Time(\texttt{S1}) + Time(\texttt{S4})$$

Similarly, `S2` and `S3` are executed for `b = 1`. Thus,

$$Time(IfStmt_1; IfStmt_2)_{\texttt{b=1}} = Time(\texttt{b == 0}) + Time(\texttt{b == 1}) + Time(\texttt{S2}) + Time(\texttt{S3})$$

The execution time estimate is set to the maximum of $Time(IfStmt_1; IfStmt_2)_{\texttt{b=0}}$ and $Time(IfStmt_1; IfStmt_2)_{\texttt{b=1}}$. Both of these quantities are lower than the estimate computed by using the default timing schema rules. Thus, by taking the maximum of these two quantities we will get a tighter estimate than by applying the vanilla timing schema rules.

Partitioning the program inputs and obtaining the WCET for each input partition is a very simple, yet powerful, idea. Even though it has been employed for execution time analysis and energy optimization in the context of timing schema [24, 25], we can plug this idea into other WCET calculation methods as well. The practical difficulty in employing this idea is, of course, computing the input partitions in general. In particular, Gheorghita et al. [25] mention the suitability of the input partitioning approach for multimedia applications performing video and audio decoding and encoding; in these applications there are different computations for different types of input frames being decoded and encoded. However, in general, it is difficult to partition the input space of a program so that inputs with similar execution time estimates get grouped to the same partition. As an example, consider the insertion sort program where the input space consists of the different possible ordering of the input elements in the input array. Thus, in an n-element input array, the input space consists of the different possible permutations of the array element (the permutation $a[1], a[3], a[2]$ denoting the ordering $a[1] < a[3] < a[2]$). First, getting such a partitioning will involve an expensive symbolic execution of the sorting program. Furthermore, even after

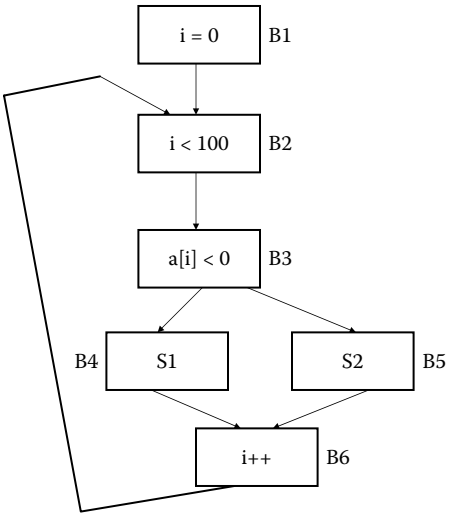


FIGURE 1.6 Example control flow graph.

we obtain the partitioning we still have too many input partitions to work with (the number of partitions for the sorting program is the number of permutations, that is, $n!$). In the worst case, each program input is in a different partition, so the WCET estimation will reduce to exhaustive simulation.

A general approach for exploiting infeasible path information in tree-based WCET calculation has been presented in [61]. In this work, the set of all paths in the control flow graph (taking into account the loop bounds) is described as a regular expression. This is always possible since the set of paths in the control flow graph (taking into account the loop bounds) is finite. Furthermore, all of the infeasible path information given by the user is also converted to regular expressions. Let *Paths* be the set of all paths in the control flow graph and let I_1, I_2 be certain infeasible path information (expressed as a regular expression). We can then safely describe the set of feasible paths as $Paths \cap (\neg I_1) \cap (\neg I_2)$; this is also a regular expression since regular languages are closed under negation and intersection. Timing schema now needs to be employed in these paths, which leads to a practical difficulty. To explain this point, consider the following simple program fragment.

```
for (i=0; i <100; i++){
                if (a[i] < 0) { S1;}
                else { S2;}
}
```

We can draw the control flow graph of this program and present the set of paths in the control flow graph (see Figure 1.6) as a regular expression over basic block occurrences. Thus, the set of paths in the control flow graph fragment of Figure 1.6 is

$$B1(B2B3B4B6 + B2B3B5B6)^{100}$$

Now, suppose we want to feed the information that the block B4 is executed at least in one iteration. If $a[i]$ is an input array, this information can come from our knowledge of the program input. Alternatively, if $a[i]$ was constructed via some computation prior to the loop, this information can come from our understanding of infeasible program paths. In either case, the information can be encoded as the regular expression $\neg B1(B2B3B5B6)^* = \Sigma^* B4 \Sigma^*$, where $\Sigma = \{B1, B2, B3, B4, B5, B6\}$ is the set of all basic blocks. The set of paths that the WCET analysis should consider is now given by

$$B1(B2B3B4B6 + B2B3B5B6)^{100} \ \cap \ \Sigma^* B4 \Sigma^*$$

The timing schema approach will now remove the intersection by unrolling the loop as follows.

$$B1(B2B3B4B6)(B2B3B4B6 + B2B3B5B6)^{99} \cup$$
$$B1(B2B3B4B6 + B2B3B5B6)(B2B3B4B6)(B2B3B4B6 + B2B3B5B6)^{98} \cup$$
$$B1(B2B3B4B6 + B2B3B5B6)^2(B2B3B4B6)(B2B3B4B6 + B2B3B5B6)^{97} \cup \ldots$$

For each of these sets of paths (whose union we represent above) we can employ the conventional timing schema approach. However, there are 100 sets to consider because of unrolling a loop with 100 iterations. This is what makes the exploitation of infeasible paths difficult in the timing schema approach.

1.2.2.6 Exploiting Infeasible Path Information in ILP-Based WCET Calculation

Finally, we discuss how infeasible path information can be exploited in the ILP-based approach for WCET calculation. As mentioned earlier, the ILP-based approach is the most widely employed WCET calculation approach in state-of-the-art WCET estimation tools. The ILP approach reduces the WCET calculation to a problem of optimizing a linear objective function. The objective function represents the execution time of the program, which is maximized subject to flow constraints (in the control flow graph) and loop bound constraints. Note that the variables in the ILP problem correspond to execution counts of control flow graph nodes (i.e., basic blocks and edges).

Clearly, integrating infeasible path information will involve encoding knowledge of infeasible program paths as additional linear constraints [49, 68]. Introducing such constraints will make the WCET estimate (returned by the ILP solver) tighter. The description of infeasible path information as linear constraints has been discussed in several works. Park proposes an information description language (IDL) for describing infeasible path information [62]. This language provides convenient primitives for describing path information through annotations such as *samepath(A,C)*, where A, C can be lines in the program. This essentially means than whenever A is executed, C is executed and vice versa (note that A, C can be executed many times, as they may lie inside a loop). In terms of execution count constraints, such information can be easily encoded as $N_{B_A} = N_{B_C}$, where B_A and B_C are the basic blocks containing A, C, and N_{B_A} and N_{B_C} are the number of executions of B_A and B_C.

Recent work [e.g., 20] provides a systematic way of encoding path constraints as linear constraints on execution counts of control flow graph nodes and edges. In this work, the program's behavior is described in terms of "scopes"; scope boundaries are defined by loop or function call entry and exit. Within each scope, the work provides a systematic syntax for providing path information in terms of linear constraints.

For example, let us consider the control flow graph schematic denoting two if-then-else statements within a loop shown in Figure 1.7. The path information is now given in terms of each/all iterations of the scope (which in this case is the only loop in Figure 1.7). Thus, if we want to give the information that blocks $B2$ and $B6$ are always executed together (which is equivalent to using the *samepath* annotation described earlier) we can state it as $N_{B_2} = N_{B_6}$. On the other hand, if we want to give the information that B2 and B6 are never executed together (in any iteration of the loop), this gets converted to the following format

$$\text{for each iteration } N_{B_2} + N_{B_6} \leq 1$$

Incorporating the number of loop iterations in the above constraints, one can obtain the linear constraint $N_{B_2} + N_{B_6} \leq 100$ (assuming that the loop bound is 100). This constraint is then fed to the ILP solver along with the flow constraints and loop bounds (and any other path information).

In conclusion, we note that the ILP formulation for WCET calculation relies on aggregate execution counts of basic blocks. As any infeasible path information involves sequences of basic blocks, the encoding of infeasible path information as linear constraints over aggregate execution counts can lose information (e.g., it is possible to satisfy $N_{B_2} + N_{B_6} \leq 100$ in a loop with 100 iterations even if B_2 and B_6 are executed together in certain iterations). However, encoding infeasible path information as linear constraints provides a safe and effective way of ruling out a wide variety of infeasible program flows. Consequently, in most existing WCET estimation tools, ILP is the preferred method for WCET calculation.

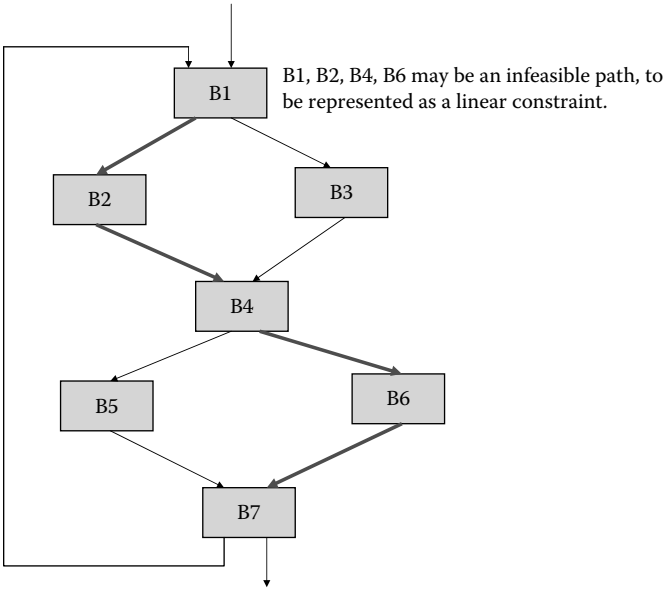


FIGURE 1.7 A control flow graph fragment for illustrating infeasible path representation.

1.3 Micro-Architectural Modeling

The execution time of a basic block B in a program executing on a particular processor depends on (a) the number of instructions in B, (b) the execution cycles per instruction in B, and (c) the clock period of the processor. Let a basic block B contain the sequence of instructions $\langle I_1, I_2, \ldots, I_N \rangle$. For a simple micro-controller (e.g., TI MSP430), the execution latency of any instruction type is a constant. Let $latency(I_i)$ be a constant denoting the execution cycles of instruction I_i. Then the execution time of the basic block B can be expressed as

$$time(B) = \left(\sum_{i=1}^{N} latency(I_i) \right) \times period \tag{1.1}$$

where *period* is the clock period of the processor. Thus, for a simple micro-controller, the execution time of a basic block is also a constant and is trivial to compute. For this reason, initial work on timing analysis [67, 73] concentrated mostly on program path analysis and ignored the processor architecture.

However, the increasing computational demand of the embedded systems led to the deployment of processors with complex micro-architectural features. These processors employ aggressive pipelining, caching, branch prediction, and other features [33] at the architectural level to enhance performance. While the increasing architectural complexity significantly improves the average-case performance of an application, it leads to a high degree of timing unpredictability. The execution cycle $latency(I_i)$ of an instruction I_i in Equation 1.1 is no longer a constant; instead it depends on the execution context of the instruction. For example, in the presence of a cache, the execution time of an instruction depends on whether the processor encounters a cache hit or a cache misses while fetching the instruction from the memory hierarchy. Moreover, the large difference between the cache hit and miss latency implies that assuming all memory accesses to be cache misses will lead to overly pessimistic timing estimates. Any effective estimation technique should obtain a safe but tight bound on the number of cache misses.

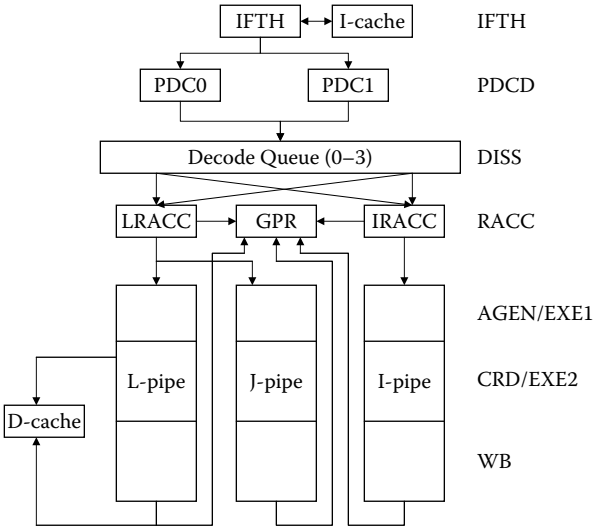


FIGURE 1.8 The IBM PowerPC 440 CPU Pipeline.

1.3.1 Sources of Timing Unpredictability

We first proceed to investigate the sources of timing unpredictability in a modern processor architecture and their implications for timing analysis. Let us use the IBM PowerPC (PPC) 440 embedded core [34] for illustration purposes. The PPC 440 is a 32-bit RISC CPU core optimized for embedded applications. It integrates a superscalar seven-stage pipeline, with support for out-of-order issue of two instructions per clock to multiple execution units, separate instruction and data caches, and dynamic branch prediction.

Figure 1.8 shows the PPC 440 CPU pipeline. The instruction fetch stage (IFTH) reads a cache line (two instructions) into the instruction buffer. The predecode stage (PDCD) partially decodes at most two instructions per cycle. At this stage, the processor employs a combination of static and dynamic branch prediction for conditional branches. The four-entry decode queue accepts up to two instructions per cycle from the predecode stage and completes the decoding. The decode queue always maintains the instructions in program order. An instruction waits in the decode queue until its input operands are ready and the corresponding execution pipeline is available. Up to two instructions can exit the decode queue per cycle and are issued to the register access (RACC) stage. Instruction can be issued out-of-order from the decode queue. After register access, the instructions proceed to the execution pipelines. The PPC 440 contains three execution pipelines: a load/store pipe, a simple integer pipe, and a complex integer pipe. The first execute stage (AGEN/EXE1) completes simple arithmetics and generates load/store addresses. The second execute stage (CRD/EXE2) performs data cache access and completes complex operations. The write back (WB) stage writes back the results into the register file.

Ideally, the PPC 440 pipeline has a throughput of two instructions per cycle. That is, the effective latency of each individual instruction is 0.5 clock cycle. Unfortunately, most programs encounter multiple pipeline hazards during execution that introduce bubbles in the pipeline and thereby reduce the instruction throughput:

Cache miss: Any instruction may encounter a miss in the instruction cache (IFTH stage) and the load/store instructions may encounter a miss in the data cache (CRD/EXE2 stage). The execution of the instruction gets delayed by the cache miss latency.

Data dependency: Data dependency among the instructions may introduce pipeline bubbles. An instruction I dependent on another instruction J for its input operand has to wait in the decode queue until J produces the result.

Control dependency: Control transfer instructions such as conditional branches introduce control dependency in the program. Conditional branch instructions cause pipeline stalls, as the processor does not know which way to go until the branch is resolved. To avoid this delay, dynamic branch prediction in the PPC 440 core predicts the outcome of the conditional branch and then fetches and executes the instructions along the predicted path. If the prediction is correct, the execution proceeds without any delay. However, in the event of a misprediction, the pipeline is flushed and a branch misprediction penalty is incurred.

Resource contention: The issue of an instruction from the decode queue depends on the availability of the corresponding execution pipeline. For example, if we have two consecutive load/store instructions in the decode queue, then only one of them can be issued in any cycle.

Pipeline hazards have significant impact on the timing predictability of a program. Moreover, certain functional units may have variable latency, which is input dependent. For example, the PPC 440 core can be complemented by a floating point unit (FPU) for applications that need hardware support for floating point operations [16]. In that case, the latency of an operation can be data dependent. For example, to mitigate the long latency of the floating point divide (19 cycles for single precision), the PPC 440 FPU employs an iterative algorithm that stops when the remainder is zero or the required target precision has been reached. A similar approach is employed for integer divides in some processors. In general, any unit that complies with the IEEE floating point standard [35] introduces several sources for variable latency (e.g., normalized versus denormalized numbers, exceptions, multi-path adders, etc.).

A static analyzer has to take into account the timing effect of these various architectural features to derive a safe and tight bound on the execution time. This, by itself, is a difficult problem.

1.3.2 Timing Anomaly

The analysis problem becomes even more challenging because of the interaction among the different architectural components. These interactions lead to counterintuitive timing behaviors that essentially preclude any compositional analysis technique to model the components independently.

Timing anomaly is a term introduced to define the counterintuitive timing behavior [54]. Let us assume a sequence of instructions executing on an architecture starting with an initial processor state. The latency of the first instruction is modified by an amount Δt. Let ΔC be the resulting change in the total execution time of the instruction sequence.

Definition 1.2
A timing anomaly is a situation where one the following cases becomes true:

- $\Delta t > 0$ *results in* $(\Delta C > \Delta t)$ *or* $(\Delta C < 0)$
- $\Delta t < 0$ *results in* $(\Delta C < \Delta t)$ *or* $(\Delta C > 0)$

From the perspective of WCET analysis, the cases of concern are the following: (a) The (local) worst-case latency of an instruction does not correspond to the (global) WCET of the program (e.g., $\Delta t > 0$ results in $\Delta C < 0$), and (b) the increase in the global execution time exceeds the increase in the local instruction latency (e.g., $\Delta t > 0$ results in $\Delta C > \Delta t$). Most analysis techniques implicitly assume that the worst-case latency of an instruction will lead to safe WCET estimates. For example, if the cache state is unknown, it is common to assume a cache miss for an instruction. Unfortunately, in the presence of a timing anomaly, assuming a cache miss may lead to underestimation.

1.3.2.1 Examples

An example where the local worst case does not correspond to the global worst case is illustrated in Figure 1.9. In this example, instructions `A, E` execute on functional unit 1 (`FU1`), which has variable latency. Instructions `B, C,` and `D` execute on `FU2`, which has a fixed latency. The arrows on the time line show when each instruction becomes ready and starts waiting for the functional unit. The processor

Random documents with unrelated content Scribd suggests to you:

the Gulf of Mexico about a hundred miles south of New Orleans," explained Jack. "You will see from this, Bill, that there are other rivers in our United States besides the noble Hudson."

Presently the train ran right along side of the great river. Bill took one look at the installment of scenery which lay spread out before them as flat as a board and then he burst out into a long and loud cackle, making, according to Jack's way of thinking, a holy show of them both.

"Why the big noise?" questioned Jack in a sour voice, for he was exasperated beyond all measure at this unseemly conduct of his pal.

"It's enough to make a bucking broncho laugh. The Mississippi eh? and you'd put it in the same class with the Hudson? Why it's nothin' but a stream o' mud," Bill made answer.

"You must remember that we're a thousand miles from its delta," expostulated Jack.

"That's nothin'; the Hudson's so wide at Noo York the politicians can't get enough money together at one time to build a bridge acrost it, see Buddy?"

And let it be said in Bill's behalf that that part of the Mississippi which is visible to the eye where the Burlington railway parallels it does make a mighty poor showing.

The boys were conspicuous for their silence all the rest of the way to St. Paul for Bill had made up his mind that he wouldn't let even his pal run down his Hudson River, and Jack had taken a mental vow that, pal or no pal, he would never again point out any wonder, ancient or modern, whether produced by nature or fashioned by the hand of man again to Bill, because the latter always pooh-poohed everything unless it was in or intimately associated with the city of Bagdad-on-the-Hudson.

As the train was nearing Livingstone, Montana, late in the afternoon of the following day the boys had entirely forgotten that the muddy waters of the Mississippi had been the innocent cause of making them a little sore at each other and all was to the merry with them again.

Livingstone is the junction where the change is made for Gardiner, the "gateway of the Yellowstone," and everybody in the car was talking about the hot-springs, the geysers, the 'Devil's Paint Pot,' 'Hell's Half-Acre' and other wonders to be seen there. Moreover quite a number of passengers were tourists who had made this long western trip for the express purpose of seeing the Park.

"We should by all means have seen the Park since we are so near it. It was a great mistake of mine to have bought our tickets through to Seattle without a stop-over here," said Jack who was genuinely regretful that he had not thought of it at the time, but it was too late now.

"Never youse mind," bolstered up Bill cheerily, "we'll stop off when we comes back and we'll have all the time we needs and plenty o' coin to do it on."

"That *listens* all right too but I have observed it is very seldom indeed that a fellow ever returns over the same trail that he sets out on, and that the time to see a thing is when he passes by the first time. Well, we'll get the gold we're after and then I'm going to make a tour of the world strictly for pleasure."

"I'm with youse Jack," responded Bill heartily.

Jack made no reply for he could see himself carrying Bill along as a piece of excess baggage and having him size up everything they saw using *his* Noo York, as he calls it, as a yard-stick to measure it by. Bill was all right for a trip of any kind where a sure-shot and brute-force were needed but on a pleasure trip around the world—well, he preferred to go it alone.

Came the time when the shine porter indicated his desire to brush off the boys and they knew that they were getting close to the end of the first leg of their journey—Seattle. They were right glad to get off the train, though withal they had had a pleasant journey and had met a number of interesting people. Among them was a Mr. Rayleigh who was accompanied by his very charming daughter Miss Vivian.

Jack had told the Rayleighs a little of his varied experiences in the World War, of his expedition to the Arctics, of his more recent journey to Mexico (giving Bill all the credit of their adventures there)

and of their proposed trip to Alaska to find gold. The net result of it all was that the chance acquaintance ripened into a warm friendship before they left the train at Seattle and his new found friends gave Jack a very cordial invitation to visit them in Chicago when he returned from his quest in the Northland, but they left poor Bill out in the cold.

Jack didn't blame Mr. Rayleigh much for he didn't know Bill's heart and he judged him by exterior appearances only. Poor Bill! the only way he could ever get a look-in anywhere was when some one saw him in action, and if Mr. Rayleigh could have seen him swatting German U-boats, or on the 'dobe in that fight with Lopez's gang he would have welcomed him with open arms.

As it was, Jack accepted the invitation so cordially given, with avidity, for he liked Miss Vivian—she was so different from those New York girls (but hush! it would never do to voice this thought in Bill's hearing or there would be a pitched battle on the spot) and she seemed to him more like a beautiful dream picture than a real being who lived in a world of three dimensions.

"Yes," he said to himself, "I've simply got to get that gold now, there's no two ways about it."

Seattle, so named after old Chief Seattle, an Indian who was friendly to the whites, is built on a site where a handful of Indians once had their village, but it was an important place even then in virtue of its being a convenient point where every once in a while thousands of Indians would meet and hold their pow-wows.

It was settled by the pale faces about seventy years ago and when the gold stampede for the Klondike was on, it was the great center for outfitting the prospectors. Later on Skagway became the chief outfitting station but as the latter town is in Alaska a duty must also be paid by those who cross over the boundary line into the Yukon Territory since it is a part of Canada. To get around this the boys concluded that they would wait until they got to Circle City and outfit up there if this was possible.

Jack was rather surprised to find that Seattle was a fine, up-to-date city in every sense of the word but of course Bill couldn't see it that way at all, so listen to him yawp:

"Youse could sot the whole blinkin' town down on the East Side of Noo York and then where'd it be? Youse couldn't find it, see!"

By the following Monday the boys had seen everything that Seattle and the surrounding country had to offer but the only things that interested Bill were the Siwash Indians and Mount Ranier.

"I suppose you'll say that the New Yorkers are dirtier than these Siwashes and that Mount Ranier can't hold a candle to the Palisades," Jack bantered him.

"Somebody must have taken the *wash* out of them Siwashes from the way they smell, and as for Mount Ranier, I'll say it's a real mountain. Let's climb it, what say, Jack?"

"After we get the gold," was his pal's comeback.

The five days that followed on the *S. S. Princess Alice* were long, bright, glorious, tiresome ones and the boys would have enjoyed every minute of the time if that disconcerting, maddening, magic word *gold* had not kept burning in their brains. They saw yellow and the nearer they came to that wonderful land in the far north, which the discoveries of gold had made as famous as diamonds have made the Kimberly mines or watered stock has made Wall Street, their very beings seemed to be transmuted into the precious metal.

Hence, neither the great Coast Range Mountains nor the wonderful glaciers appealed overmuch to these youngsters who had set their hearts on getting gold out of the Yukon-Arctic district just as firmly as had ever the most seasoned prospector.

But Juneau did make an impression on Bill for he heard tales of gold up there the like of which he had never heard before. Only once did he think to belittle the town by making odious comparisons of it with his "Noo York" but with Jack's help he smothered the attempt for he was in the gold country now and was carried away by that malignant disease known as the *gold fever*.

CHAPTER III
ON THE EDGE OF THINGS

The *Princess Alice* made a stop for a few hours at Juneau, a town standing on a promontory between Lynn Canal and the Taku River, and the boys, with many other passengers, disembarked to see what they could see. Here for the first time they felt they were getting pretty close to the field of their future activities for they were in Alaska, the land of the midnight sun and the aurora borealis, the moose and the caribou, the prehistoric glaciers and—hidden gold.

Across the water a great mill was in full blast and as they stood looking at it a big, grisly sort of a man, who appeared to be between fifty and sixty, and whose clothes showed that he was an old time prospector, moved over toward them. Evidently he had in mind the idea of holding some small conversation with them, for up on top of the world the inhabitants do not consider formal introductions as being at all necessary when they feel like talking to any one.

"Goin' to buy it boys?" he asked, grinning good-naturedly to show that his intentions were of the best.

"Afore we do, we'd kinda like to know what it is, for we'd hate to buy a pig-in-a-poke," replied Bill smiling just as cheerfully, only, as I have previously mentioned, whenever Bill smiled the scar across his cheek made him look as if he was getting ready to exterminate a greaser.

"Oh, I see, you youngsters are new up here—tourists maybe," came from the big throated man.

"We're new up here all right," admitted Jack, "but we're not up here to see the sights, or for our health either, but to do a bit of prospecting."

"Shake pards," and he held out a calloused hand, as big as a ham and as horny as a toad's back, to each of them in turn. "I'm Hank Dease, but in these parts I'm known as Grizzly Hank. And who might you fellows be?"

"I'm Jack Heaton of New Jersey, and this is my side-kick, Bill Adams of New York City, New York County and New York State, and there with the goods as needed."

"I blazes! I'm right glad to know you boys," drawled Grizzly Hank, "for you look to me as if you're made o' the right kind o' timber. Since you're strangers here I'll tell you about Juneau, which I allow is the finest city in the world."

Now Juneau has a population of about two thousand people, so, naturally, Bill was going to jump right in and monopolize things by asking Grizzly Hank if he'd ever been in Noo York, but Jack gave him the high-sign not to break in and so for once his pal held his peace.

"I'll tell you about the wonderful things we have here first and then if there's any little thing you want to know about prospectin' up here or in the Yukon Territory I'll tell you as good as I know. I've been in this country for nigh onto thirty years and you see how well I've panned out, but you fellows may do better—a few do, but, I blazes, most of 'em don't."

Grizzly Hank had found a couple of good listeners and as he liked to talk he was making the most of them while they lasted.

"That's the Treadwell mill you are lookin' at over yonder on Douglas Island. It has an output of gold that runs upwards of eighty thousand dollars a month. The first gold ever found in Alaska was down at Sitka in 1873, but it was old Joe Juneau, a French-Canadian prospector, who showed that gold could be mined here in payin' quantities.

"At that time another prospector named Treadwell who was in this district had loaned a little money on some claims over there and finally had to take them for the debt. Later on he bought French Pete's claim which lay next to it for the magnificent sum of five hundred dollars; and these claims which he bought for a mere song are the great Treadwell mines of to-day. I blazes! There are some

other mines in this district and since Treadwell took over the original claims the output of gold has been to the tune of a hundred million dollars and the end is nowhere yet in sight. I blazes!"

"Do you mean to say, Mister Dease, that gold is mined over there like coal?" asked Bill, thereby exposing his ignorance.

The grisly prospector looked amused but he recalled the time when his own ideas of mining gold had been just about as vague.

"You see, boys, gold is found in several ways up here. Sometimes it is 'bedded in quartz when the *ore*, as it is called, has to be mined and then crushed in a stamp mill to get the gold out; more often it is found as free gold, dust and grains and bits of pure gold mixed with the dirt when it must be *panned*, that is, put in a pan and the dirt washed away and then the gold, which is the heaviest, falls to the bottom of the pan, and again," he lowered his voice to make what he was about to tell them more impressive, "nuggets of gold are picked up from bits the size of a pea to chunks as large as my fist! I blazes! It all depends on the locality."

"These diggin's here are quartz mines and the ore is of mighty low grade—only a couple of dollars in gold to the ton of quartz. To get this gold out the quartz, or ore, is crushed in a mill called a stamp, and the Treadwell has the largest number of stamps of any mill in the world—upwards of two thousand, I blazes!"

Grizzly Hank paused for a moment to get a fresh start.

"Go on Mister Hank, we're listenin' with both ears," urged Bill.

"As you were saying—" Jack paced him.

"As I was about to say," continued the prospector, who was every whit as appreciative of his audience as it was of him, "when Treadwell began to take out gold, old timers all along the coast clear down as far as 'Frisco heard of it, came up and pushed further north believing that they would find other lodes of gold bearing ore and they believed right, I blazes!

"That other mine over there on Douglas Island that you see to the right is the Mexican Mine but it's small fry as against the Treadwell for it only has a hundred and twenty stamps working."

"We're not pertiklarly keen on Mexican mines, oil wells or anything else that goes by the name of *Mex*—we had all the Mexican stuff we wanted when we was down there six months ago," broke in Bill to whom the word brought no very pleasant recollections.

"To this side of the Mexican mine," went on the prospector, "is the *Ready Bullion* mine and it has a two hundred stamp mill."

"*Ready Bullion* listens good to me," admitted Jack, once more breaking into his discourse.

"Shortly after the Treadwell mine began to show itself a bonanza, a story went the rounds that it was an accidental lode, or a *blowout* as we call it; that is, it was a lode of gold deposited there by some gigantic upheaval of the earth when Alaska was in the makin' and that it was the only place north of fifty-six where gold could be mined at a profit.

"I always believed that yarn was set agoin' to keep other prospectors out of the country; but when it kept on producin', men with picks and shovels came here just the same, and what happened was that other deposits were found and these are the mines that are bein' worked now in southern Alaska.

"Still other prospectors pushed on further north with their packs on their backs, on sleds which they pulled themselves or which were hauled by dog teams, on horses and mules, and they toiled up the *Trail of Heartache*, as the nearly straight-up *White Pass* trail was called in those days. I blazes, *and*, I was one of 'em.

"Once on the other side of yonder range we prospected for gold bearin' quartz, and panned the river beds until we reached the Klondike River. There is where Carmack, with two Indian pards, Skookum Jim and Tagish Charlie, had already staked rich claims. One day Carmack went down to the stream to wash a piece of moose he had killed and it was then that he saw gold in the water and when he panned it he got more nuggets than his eyes could believe. News of gold travels faster than greased lightnin' and it was not long before the biggest gold stampede was on that ever took place in the golden history of gold! I blazes!

"Over night the Klondike became famous and wherever human bein's lived that spoke a language it was a word that they knew and it meant but one thing to them—and that was gold. And, I blazes, the world knew that gold was bein' panned out in the Klondike by hundreds and thousands and hundreds of thousands of dollars and the world went crazy over it.

"When I got there one mornin' I was dead-broke but by night I was a rich man. It was nothin' to wash a hundred, five hundred, I blazes, a thousand dollars from a few pans of gravel. And still further north, somewhere along the Porcupine River, Thornton and a couple of his pards discovered a blow-out where nuggets of gold were so thick they could pick 'em up like stones; they packed them in moosehide sacks and corded them up like stovewood until they had all the gold they thought they could carry out of the country."

Grizzly Hank had the boys going for fair. They stood as though they were magnetized to the spot. Both were itching for more detailed information but neither spoke his mind for they had agreed before they left New York that while they would have to admit they were prospectors bent on finding gold, like countless thousands before them, they would give no hint, under any circumstance, of their real mission to any one.

"Go on—" said Bill impatiently.

"Yes, pards," he went on, his sharp, deep-set eyes brightening which showed that however it was he had failed to keep the elusive metal he had found, his long quest left no cause for regret; "yes pards, the gold belt runs from the Gulf of Alaska to the Arctic Ocean, and the further north you go the more gold you'll find and—the harder it will be to get it *down under*.[2] I'm goin' to the Porcupine River district as soon as I can get some one to grub-stake me——"

2 In Alaska and the far north the United States is called *down under*.

A mighty bellowing blast came from the triple throated whistle of the steamer at the dock and drowned out the alluring voice of the prospector pioneer. Then the warning sound subsided for a moment.

"There's your boat a-whistlin' an' if you're goin' on her you'd better scoot. I blazes! Good-by and good luck."

They started for the boat on the run but their minds were in a semi-torpid condition, for the old miner had surely enough set them by the ears. When they were again on the deck of the *Princess Alice* and had somewhat recovered from the magic of his words they fell to discussing gold, Grizzly Hank and a few other consequential things.

"Moosehide sacks of gold corded up like stovewood!" repeated Bill blinking his blue eyes.

"The farther north you go the more gold you'll find!" reiterated Jack, for the words sounded like ready money to him.

"Shake, old pard, we're on the right trail," and the boys struck hands with a vengeance. "I was thinkin' as how we orter have taken Grizzly Hank along with us," commented Bill; "he knows all the ropes and he'd a-come in mighty handy."

"I thought of that too when he was talking to us but then we'd have to split up our winnings into thirds which would mean that we'd simply short-change ourselves out of a couple of million dollars or so. Then again his ideas and ours would probably be entirely different for he's a prospector of the old school while we are discoverers of the new school. Finally, 'two's company and three's none' is just as true, I imagine, of the trail as it is of a parlor date."

"Agreed to on all points," said Bill, "but when we comes back let's grub-stake him to the limit so that he can eke out a million or so on his own account afore he kicks-in."

Skagway was the jumping off place as far as the *Princess Alice* was concerned and the boys were right glad of it for they were anxious more than ever to get into the heart of things. The town is on the Chilkat Inlet at the head of Lynn Canal and, like many others along the coast, it has a mountain for a background.

They stopped over night at Mrs. Pullen's hotel, which is also a wonderful Alaskan museum, and as they were looking about they came across a rack of the inevitable picture post cards. Bill said he was of a mind to send one down under to a certain little telephone countess, (whom he could see in his mind's eye masticating the indestructible listerated nuggets and hear her say in the deep

recesses of his auditory organ "who do you want to talk to?" with the "smile that wins.")

On one of the post cards was a picture of a very pleasant, mild mannered looking gentleman whose kindly eyes and benevolent mouth bore out Jack's statement that all men north of fifty-six are *white* at heart. Under the picture on the card of the somewhat incongruous caption of *Soapy Smith*.

"I suppose he's the Sunday School Superintendent, owner of the First National Bank and mayor of this burg," Bill remarked to his partner.

A prosperous looking individual standing near-by overheard Bill's facetious comment, smiled sadly and said:

"I take it you boys haven't heard the story of Soapy Smith and so I'll enlighten you as to the manner of man he was. Soapy came by his saponified cognomen honestly for he began his career as a full member of the fraternity of gentle grafters. Soapy's line was to wrap up a ten dollar bill with a small bar of soap and sell it from the tail end of a wagon for the small sum of one dollar.

"Then the lamb would take his purchase around in the back alley where no one could see him, and open it up and then he would find that he was out just ninety-nine cents, for while he had the soap the slippery ten-spot still remained as a part of Soapy's financial reserve fund.

"But this graft was too legitimate for Soapy for he had to give a bar of soap worth at least a cent to each and every purchaser. Having accumulated a little coin he drifted in here with the stampeders in '98 and opened up a saloon, dance-hall and gambling house. As if this game was too honest he organized a gang of outlaws and they robbed men and killed them too, right and left.

"Law abiding citizens got tired of these hold-ups, for the prospectors and miners began to go through Dyea and use the Chilcoot Pass rather than take a chance of meeting Soapy and his gang in Skagway or on the White Pass trail. So a *Vigilance Committee* was organized and at one of their meetings one night

they put Frank Reed at the gate to keep Soapy and the members of his gang out.

"As soon as Soapy heard of the meeting he took his shootin' irons and went over to it where Reed promptly refused to admit him. Came two simultaneous pistol shots; Soapy fell dead and Reed lived for a couple of weeks and then he cashed in. If you go up to the canyon you'll see the graves of both these men in the cemetery there. So you see you can't most always tell by lookin' at a man what is under his vest."

The next morning the boys took the train for White Horse, about a hundred and ten miles due north at which point they would make connections with a boat on the Yukon River. While the stampeders had toiled up the icy trail of White Pass, their backs breaking under their packs and their hearts breaking under the torture of it all, the boys were now making the trip in a comfortable train of the White Pass and Yukon Railway, the first in Alaska and the Yukon Territory.

"Isn't just exactly like ridin' on the *Twentieth Century*, is it Jack?" observed Bill as the train crept at a snail's pace up to the summit.

Just then the train rounded a curve blasted out of solid rock and they looked straight down a thousand feet into a canyon.

"More like a trip on the *Elevated*," suggested Jack.

Once over the Pass the engineer opened the throttle a little and the train picked up in speed. Then by way of varying the kaleidoscopic changes of scenery the train shot into a tunnel and out of it onto a tremendously high bridge that spans the Skagway River which flows tumultuously over the rocky bottom on its way to the gulf.

A few miles beyond they crossed an old wagon road which was being built to connect White Horse with White Pass but the railroad was completed first and took its place. A dozen miles or so farther on they saw some log cabins which the conductor of the train pointed out as having been the center of White Pass City, one of the tented towns that had sprung up during the mad rush to the Klondike, and when it subsided the town vanished.

Then came into view Glacier Gorge and high above it the train sped along its very edge, then wound up a long grade, when spread before them were the Sawtooth Mountains and Dead Horse Gulch.

"Sounds like the name of a dime novel I onct read," reflected Bill.

"Why Dead Horse Gulch?" Jack asked the conductor.

"Because when the rush was on in '98 thousands of the pioneers brought their horses with them and so many of them died down there from starvation and overwork that their bodies choked up the gulch.

"See that sheet of water yonder?" he continued, "that's the beginning of Lake Bennet and there the hustling, bustling, town of Bennet once was. As soon as the gold crowd from Skagway reached this lake they gave up the trail and threw together rafts and craft of every description. They piled their outfits on or in them and then floated down the Yukon River to the Klondike, unless they were drowned first, as many were. You'll be glad to know, boys, the train hesitates twenty minutes at Bennet for victuals," and the boys thought it was high time that it did so.

When this important function was over and they were again on the train it ran along the edge of the lake until the lower end of it was reached where the friendly *con* called "Carcross! Carcross!"

"This town," he told them, "is built on a place where the Indians used to watch for the caribou to cross and this is the cause why of its name."

After a short ride their rail trip—the last they would have for many, many moons—came to an end at White Horse, on the Thirty Mile River. They considered they were playing in great good luck, for the steamboats leave only twice a week for Dawson and one was scheduled to sail that night.

This gave the boys plenty of time to look around White Horse but they saw with eyes dimly for their vision was as blurred by their quest for gold as ever were those who had rushed madly through there in the days of '98.

Bill opined that he "liked White Horse fine as it has two boats a week we can get away on." As a matter of fact it is a lively town for

the steamboats take on their supplies here for their down river trips.

The boys walked over to the White Horse Rapids, as the Indians called it after a Finnlander because of his light hair and whom they thought was as strong as a horse, after he had lost his life in its swirling waters. And hundreds of other lives and dozens of outfits were lost in the wild scramble of the early prospectors to get to the gold fields.

But neither Jack nor Bill gave more than a passing thought to these foolhardy and adventurous souls who had risked and lost all in their futile attempts to get to the Klondike; much less did they think of those who had made the golden goal and won out in the finality of their efforts, for the boys' own scheme consumed every moment of their time, and all of their energies were directed upon the consummation of it since they were gold seekers just as truly as were any of those who had gone before.

The steamboat *Selkirk*, which was to carry the boys from White Horse to Circle City, was of the old time kind that was used on the Mississippi and other rivers half a century ago; that is, it was of the wood-burning, stern paddle-wheel type.

As they stood out on deck the next morning Jack tried to lose sight of the big issue for the moment and he imagined himself to be the first explorer who had traced the Yukon River in this region. If he had not had gold on the brain it would have been an easy thing to do for here were the same virgin meadows, primeval forests and silent fastnesses just as they were when the Russians laid claim to Alaska. And the gold, he reasoned, that was here then is, for the greater part, here now.

Not once since they had left Seattle had Bill compared anything with his Noo York, at least not out loud, but when they were passing through the headwaters of the Yukon he said as though he was talking to himself, "It hasn't got anything on the *Spuyten Duyvil*," which, let me elucidate, is a tidal channel that connects the Harlem River with the Hudson River and so forms the northern boundary of Manhattan Island on which New York City proper is built. But in the eight hundred and sixty odd mile trip down the Yukon to Circle City

Bill had ample opportunity to amend his snap comparison and even then he was fifteen hundred miles from its many channeled delta where it flows into the Bering Sea.

"Doesn't look much like the naked north or frozen regions that the folks back home think it is," remarked Bill, as they passed a *tundra* (pronounced toon´-dra) which was thick with grass and shrubs and sprinkled with various plants in flower.

"I'll say it doesn't," replied Jack, "but wait, we haven't run into winter weather yet."

As the boat plied its way softly and swiftly down the Yukon they saw occasional Indian villages, the men taking life easy, the children playing and the squaws busy drying the golden salmon on poles set in the sun. Then to the great delight of both boys they saw a caribou swim out from the shore intending, probably, to cross to the other side, but frightened by the modernity of the throbbing, smoking monster he swam back faster than he came, and on gaining the shore he disappeared from view.

Another time Bill went over to Jack, who was talking with some passengers, and saluting as to an officer he said, "I have to report, sir, a bear on the starboard bow." And sure enough there stood a huge bear high on the ledge of a rock and so motionless was he that he seemed carved out of the rock itself; but inwardly he was fully alive to this mechanical invasion of his eminent domain.

Never was a river trip of such wild beauty, so full of interest and yet such soothing quiet as this one the boys were now making and it would have proved doubly delightful if they had been pleasure seekers instead of gold seekers. The only breaks in the continuity of the run were made when the boat nosed its way along a bank and, finding an anchorage, she *wooded up*, that is she took on wood to be burned under her boilers.

Now the river widened and the boat ran into the more placid waters of Lake LeBarge which Jack pointed out to Bill as having been the scene of action in *The Cremation of Sam McGee*, a poem by Robert Service. On reaching the lower end of the lake the boat shot down the Thirty Mile River where the swift current winds forth

and back like a tangled rope and it takes a pilot who knows his trade to hold her to the channel.

But the most exciting piece of navigation is at Five Finger Rapids, for here the river narrows down into a neck and almost closing the latter are five ugly finger-like rocks projecting above the surface with the water swirling swiftly round them in mighty eddies. It looked to Jack and Bill as if there was not enough room for the boat to pass between any two of them but this didn't seem to worry the pilot any who held her nose hard toward the middle finger.

The boys thought that he must be tired of life. But hold there matey, just as they had timed her to strike the rock he bore down hard on his wheel to port and the boat missed the rock by the skin of its teeth, Their hearts dropped back from their throats to their thoraxes again and they believed they still stood a fair chance of finding the gold they were after.

And now comes Dawson into view—Dawson in the heart of the Klondike—the Dawson of tradition, adventure, romance and—of gold! This is the identical town where that great army of pioneer gold seekers, who braved the rigors of the winters, the dangers of the rapids, the stresses of starvation and the robbers of Soapy Smith's gang, found themselves if they were unfortunate enough to be so fortunate.

As the steamboat ties up here for half a day to load and unload its cargo the boys went on a hike over to an Indian village called *Moosehide*, a little way down the trail from Dawson. On returning to town they got the *borry*, as Bill called it, of a couple of horses and rode out eight or ten miles where some great dredges were at work bringing up the sand and gravel from the streams and hydraulicking equipments were washing the gold out of it.

"This kind of mining," Jack said to his partner, "is simply panning out gold on a big scale by machinery, and gold fields that are not rich enough to be worked profitably by a prospector will yield gold on a paying basis where hydraulicking can be taken advantage of."

"It's too slow a game for me," was Bill's idea of the scheme, "I wants to pick it up in chunks."

"That's what we're here for," Jack made answer.

They left Dawson that evening and the next morning still found them in the Yukon Territory, but shortly after breakfast the boat crossed the International boundary line and they were on good old U. S. soil again. The boat soon made a landing at Eagle City where Fort Egbert is located and the first thing Jack spied was a big wireless station which he knew belonged to the U. S. Army.

From Eagle to Circle City, or just *Circle* as it is called for short, is a sail of a hundred and ninety miles. Both Jack and Bill were dead tired of traveling and they hailed Circle as heartily as they would have hailed their own home town. But they didn't know what they were hailing. The only outstanding fact with them was that they had *arrived*, or at any rate they had gone as far as trains and boats could carry them toward the goal of their desires. The bridge was swung ashore and they got off without delay. The whistle blew a couple of sonorous blasts, and the boat backed off and went on her way down stream.

In the days of the gold rush Circle had been the great outfitting town in these parts. It was built up entirely of log cabins and it had more log cabins than any town had ever gathered together before or since. Why Circle City? Whence the name? Because when the town was started it was believed to be located right on the Arctic Circle but later it was learned that it was a good eighty miles below the Circle.

As the boys stepped ashore they were greeted by a few white men, some Indians and the ear-splitting howls of the huskies.

"I tell you Bill, we're on the very edge of things."

"You said a mouthful, pard," was that worthy's sober reply.

CHAPTER IV
WHEN BILL AND BLACK PETE MET

The boys wore sorely disappointed in Circle for while it had been, as they had heard, "the largest log house town in the world," and as far as log houses go it was yet, for that matter, still that essential moving principle that makes up a town, namely the inhabitants, was lacking.

But times have changed since the early '90's and now all that remain of its population are a few men who look after the stores and a handful of prospectors, miners, hunters and trappers who come into town to buy their supplies, and these hearten it up a bit. As for the empty log houses they serve only as so many monuments to commemorate the time when the town was alive and full of action.

You ask why the town died out? I'll tell you. Gold was discovered there in 1894 and for the next four years its growth was phenomenal —the wonder of all Alaska; but when the Klondike was opened up the inhabitants left everything behind them and made a mad rush for the new gold fields, and so at the present time there is little left to tell of the glory that was Circle's.

The way Jack had figured it out coming up on the boat was that they would get their clothes, grub, sleds and dogs at Circle, which prospectors and others he had talked with said they could do, and then when they were all fixed and winter had set in they would push on over to the land of the Yeehats and there establish a base from which they could work.

This base of supplies was to be like the hub of a great wheel the circumference of which would include all of the territory to be prospected and their local expeditions would be like the spokes, that is they would strike out with their dog teams, traveling light, taking a

new line of direction each trip they made. In this way they could, he said, make a thorough search for the hidden gold that those before them had struck so rich but which for divers reasons best known to those who had sought it had never been gotten out of the country.

His best thought, as he had previously explained in answer to an objection of Bill's, was to make this search during the winter months instead of doing it in summer-time in virtue of the fact that they could then use dog sleds and this would enable them to cover the ground without working themselves to death and do it at a goodly clip besides.

Now, when Bill had set his eyes on the deserted City of Circle he instantly took a violent dislike to it. Having become fairly well posted on the geography of *Ilasker*, as he still persisted in calling it, he concocted the notion that what they should have done was to come up in the early spring and go on by boat to Fort Yukon, which is about eighty-five miles farther on down the river.

From there, he contended, they could have gotten a couple of canoes and paddled up the Porcupine and Big Black Rivers until they were close to where the International boundary line crosses the Arctic Circle. This done, (according to Jack's own reasoning he said), they would be about as near the place where they wanted to make their winter quarters as they could get. But there was no getting away from it, they were now in Circle with winter fast coming on and it was too late to change the work sheet as previously laid out.

By the time this argument was over, the boys had reached the Grand Palace Hotel, an enormous log building of two stories of the regulation kind to be found in all frontier and mining towns.

Running nearly the length of one side of the hall as they entered it, was a bar with a hotel register on the end nearest the door. At the extreme farther end of the hall a platform had been built up about as high as a man's head, while any number of small round tables covered with worn-out and faded green cloth were strewn about the room.

The owner of the Grand Palace in the days antedating the Klondike rush was Sam Hastings, or *Silent Sam* as he was called,

because he never spoke unless he was spoken to and his replies were always pithy and to the point. His face was smooth shaven; he wore a low crowned, narrow brimmed Stetson hat, a rolling collar with a flowing tie, silk shirt with diamond set gold buttons in the cuffs, a Prince Albert coat with a six gun conveniently within reach under it, doeskin[3] breeches and kid button shoes. Unlike Soapy Smith he was honest, as men of his type went in those days, but like Soapy he died with his button shoes on.

3 *Doeskin* is a kind of fine twilled cloth much used in those days for making breeches.

Now let this close-up of Silent Sam fade away and take a look at a snap-shot of Doc Marling, the present owner of the Grand Palace and you will observe a further change that time and circumstances have wrought in Circle.

Doc is a big-headed man and bearded like a couple of pards. He wears a woolen shirt, under which beats a fair to middling heart; his breeches are also woolen tied around his ankles and he has on a pair of deerskin moccasins.

He is no shooter—you could see that the moment you look at him—but it is history up yonder that he once choked a bear to death with his hands alone.

He was the only animated object in the great bare room when the boys walked in and they felt like a couple of mavericks that had been cut out from the herd. No more lonesome place had either of them ever been in this side of Nyack-on-the-Hudson.

But Doc Marling didn't seem to feel that way, since after being there for twenty odd years perhaps he'd gotten used to it. He invited them to inscribe their names on the hotel register, after which he led the march down the hall—it seemed to the boys as if it was a block long—thence up the stair-way whose well-worn steps showed clearly that Circle had been very much alive in the days of her youth, and then to their room which was altogether too big.

"One thing sure, we'll get in practice here for the long winter that is ahead of us," reflected Jack philosophically.

"It wouldn't be half-bad if we had a 'phone connection with the *American Consolidated Oil Company* back in Noo York, but where are

we? Five thousand miles away and not even a wireless station nearer than Eagle. 'I blazes!' as Grizzly Hank down at Juneau says," groused Bill. His indisposition was curious in that no matter how strenuous the tide of battle might be he had never a word to say, but inaction always behaved as an irritant to his nervous system.

Came soon the loud jangling of a bell and they knew it for a call to supper. They followed where it led and sat down to their first meal in Circle, and it was good. There were ten or a dozen men at the table with them and up here at the very outpost of civilization, where men are what they are, they all fell into loud and easy conversation.

"We're in the hands of white men, as I said we'd be, back there in New York," Jack told his partner when they were again in their room.

Just as they were about to turn in they thought they heard a phonograph going, and as "music hath charms to soothe the savage breast" they went down into the big hall to be soothed.

While in pre-Klondike days it was of nightly occurrence to find four or five hundred people gathered in the hall, there were now congregated perhaps some twenty-five or thirty men, and these were made up of Americans, French-Canadians, Indians, half-breeds, and a Chinaman or two, to say nothing of the bear.

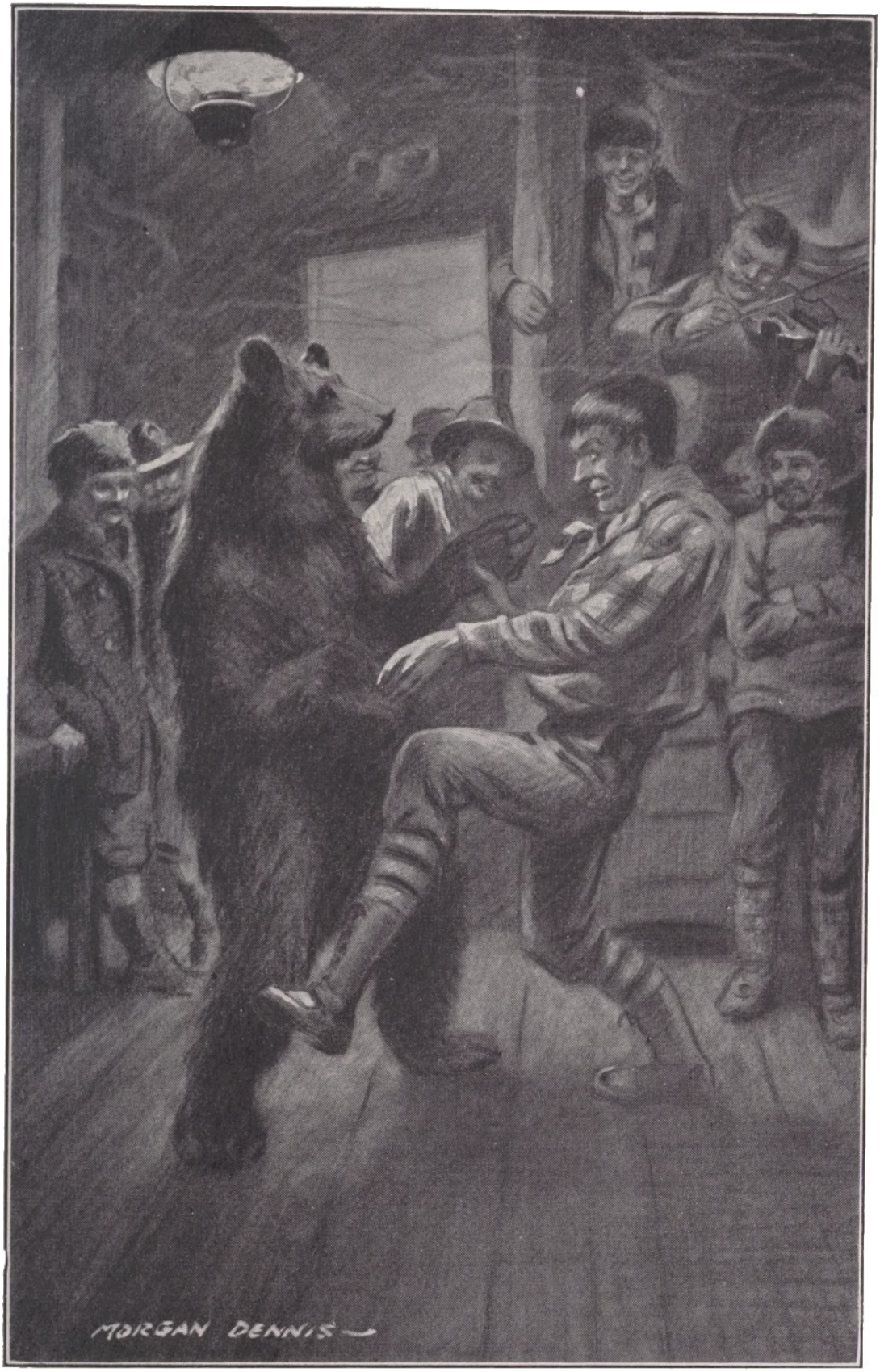
A few of those who composed this agglomeration of humanity, were the scum of the earth but most of them were men of strong character and sterling worth. Considering that they were on the very edge of things they were bound to be a rough and ready lot but taken all in all they were well behaved and peaceably inclined—all except one and he was Black Pete.

While the crowd by no means filled the void of the big hall, still it breathed enough of life into the stagnated atmosphere to take off the sharp edges of their lonesomeness.

Now instead of a phonograph they discovered that the source of the music originated in a tall, rangy miner with a big bushy mustache, who was sitting on the platform and sawing away on a fiddle as if his whole soul was in it. Near the platform some kind of a disturbance was going on around which the onlookers had formed

themselves into a ring. Whatever it was they were greatly interested and from the roars of laughter they were evidently enjoying it hugely.

Jack and Bill elbowed their way deep enough into the ring to see what the frolic was and what they saw they concluded was about as good as an act in a side-show. In a word it was a team of dancers executing with great precision and solemnity the “bear-trot”, or “bear-hug”, or “bear-something-or-other”, for a young French-Canadian and a big brown bear, who stood erect on his hind legs, when he was as tall as his keeper, were executing a most ludicrous, albeit, a lumbering sort of dance.


MORGAN DENNIS

After a spell Rip Stoneback, the fiddler, ceased scraping the catgut strings with his horse-hair bow and the trainer and his bear wound up their exhibition with a wrestling bout that tickled the everlasting daylights out of these simple northmen, from which it could be fairly deduced that, after all, they were really only boys "growed" up.

The boys mingled freely with the knots of men taking in what they had to say about everything in general and little things in particular, for it was all brand-new and novel to them. Jack struck up a conversation with a young fellow named Jim Wendle from 'Frisco who had staked a claim over on Preacher Creek.

"The boys here are all right," he was saying to Jack, "there's only one fellow who is really hard boiled and that's Black Pete over there. He's laid out every man he's ever tackled, either with his fists, or his knife and I've heard that he shot a man once. He's meaner than all get out when he's had a few drinks so don't get into any argument with him. Agree to anything he says if he talks to you."

Black Pete did not look the part of a "bad man" though his face was hard and his complexion was swarthy. He was not very tall, had tremendous shoulders and having lived in the open Northland all his life he knew the run of men who gathered here. He was thoroughly disliked in Circle because of this disposition on his part to always want to pick a fight and there were men thereabouts who were actually afraid of him.

At about the same time that Jack was getting his information concerning Black Pete another prospector was tipping off his history to Bill and it was lucky for both of the boys that they were "let in" on his past performances when they were.

Black Pete and a boon companion were leaning against the bar when the latter made some passing remark about that young stripling and his partner who had just landed in Circle.

"Sleem keed heem all right," returned Pete, "but I no got use for heem pardner—zat fellow weez da cut cross hees cheek. I give heem beeg leeking sometime. Maybe theese night. Watch a

meenute. I have som' fun with sleem keed." Black Pete called to Jack and motioned him to come over, but as the latter had not been introduced he paid no attention and this aroused Black Pete's ire. Then he and his companion started over toward Jack and Jim Wendle.

"Be careful now," his friend cautioned him.

Black Pete laid his hand on Jack's shoulder in a perfectly friendly like manner and said:

"You and Jeem com' heeva dreenk weeth me?"

At that Jack got up from the table and looked Black Pete square in the eye.

"I don't drink," he said shortly.

Black Pete was mad clear through, that much was plain.

Bill who had been taking a hand in a world-old game called *poker*, happened to see Jack and Black Pete facing each other and he divined trouble. He laid down his cards and went over where his pardner and the bad un were, to *listen in* on the conversation.

"Heeve a seegar, then," the Canuk insisted catching hold of Jack's arm and pulling him toward the bar.

Taking a firm hold on Black Pete's wrist Jack removed his hand from his arm and said, without the slightest inflexion in his voice, "I don't smoke."

Then the unexpected happened—that which had not happened in Circle in perhaps a dozen or twenty years before.

"You don't eh?" growled Black Pete, infuriated at Jack's cold refusal to join him in either one or the other, "then deem you, heeve a bullet!"

At the same time he whipped out his six-shooter and pulled the trigger, but his marksmanship was bad, for Bill had caught him by his throat from the side and pulled his body over so that the bullet crashed through the roof, instead of boring a hole through Jack's body.

Expecting that the remaining chambers would be emptied in the struggle which took place between Bill and Black Pete the crowd

dropped to the floor, jumped behind the bar, crawled under tables—all except René and he kept his trained bear between himself and the business end of the gun the bad man of Circle and the Harlem boy were struggling for.

These latter two were well matched though there was no doubt but that Black Pete who was the larger was also the stronger, but sheer brute strength could not gain the mastery where the tricks of the wrestler's art are brought to bear and Bill had a little the best of it.

As the crowd rightly guessed when the first shot was fired, Black Pete did pull the trigger every chance he got until all of his cartridges were shot off but each time the bullet that was intended for Bill went wild and neither he nor the others were scratched. One bullet, though, shivered the big plate glass mirror over the bar into a thousand pieces and Doc Marling, the proprietor, knew that he was having bad luck just then to the jig-time of three hundred dollars, even if it didn't keep on for the next seven years.

All the time the struggle was under way Jack stood by as though he was watching a friendly bout in Prof. William Adam's Academy on Manhattan Street in the good old days. More than one of the onlookers wondered why he didn't crack a bottle on Black Pete's head and so help out his partner, but this was not the way the boys did team work. In a set-to of any kind whether it was with bare knuckles, with knives or with pistols neither one would take a hand in the affair the other was engaged in unless, as Jack had once explained to me, it was "absolutely imperative."

And this status of the fray was far from having come to pass, at least that was the way Jack sized it up. The crowd must have kept count of the shots fired for when the last one took place they quickly picked themselves up from the floor, or crawled out from their safety-first hiding places, and gathered around Bill and Black Pete who were still at it.

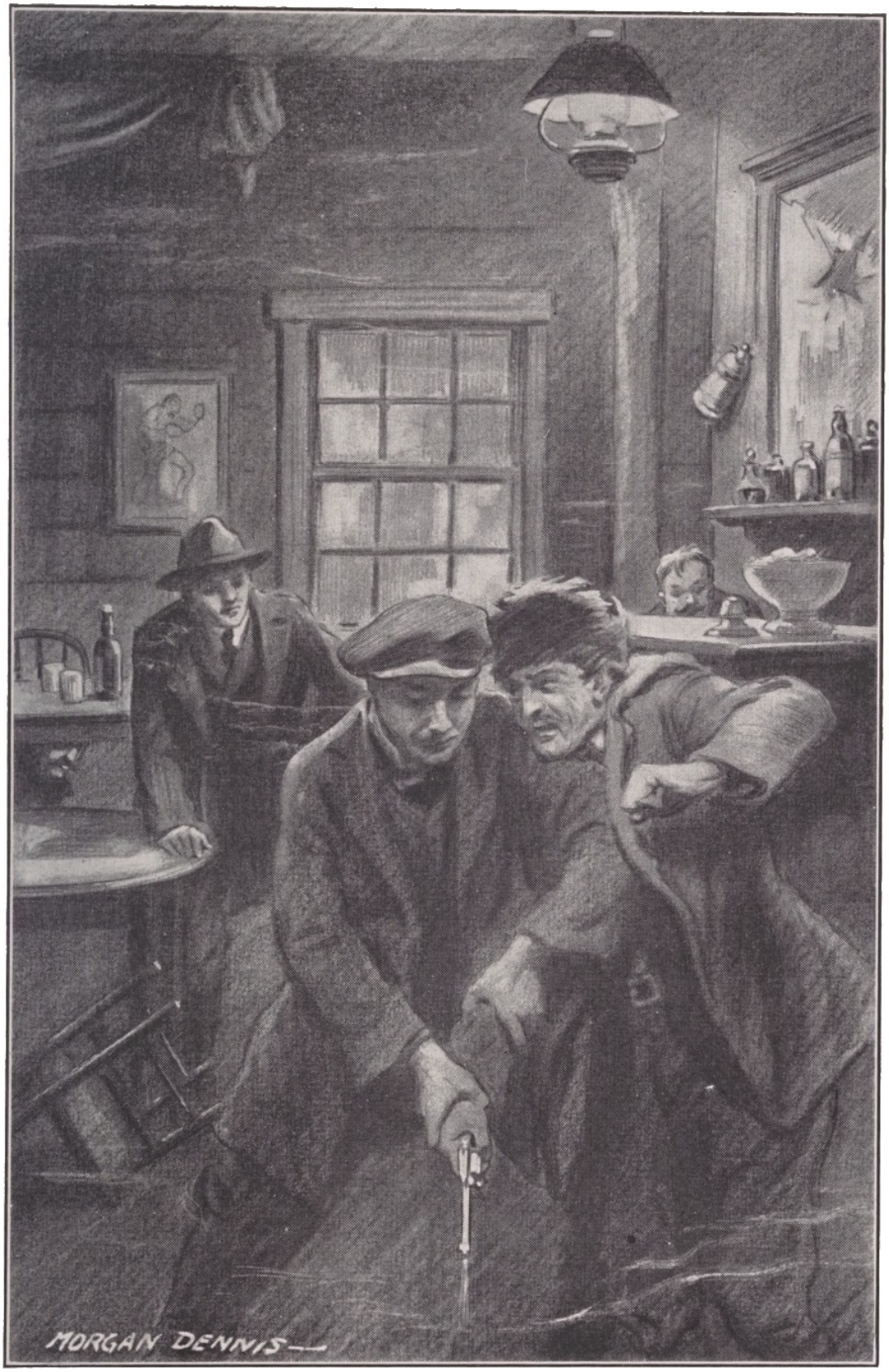
Whether it was due to the final breaking down of his courage, failing strength, too much hootch or the superior tactics of the trained athlete, was not apparent, but slowly Bill overpowered his

opponent, threw him over his shoulder, when he struck the floor on his back, and pinned him down so that he could not move. After all had seen that Black Pete was helpless Bill let him up.

There was wild cheering for the victor and some one brought Bill a big glass of forty-rod.

"You have well earned it boy and you need it," he said as he offered the glass to him.

"I never drink," said Bill and it was given instead to Black Pete to revive him again.


MORGAN DENNIS

When the latter had regained his feet, and recovered from the shock a little, he offered no explanation for his defeat, but in his deep humiliation he moved over toward the door to make as dignified an exit as he could in the quickest possible time.

"Hey, where are youse goin'," Bill called out after him. "Come back here and sit down at this table and let's be friends, for I never holds a grudge after I have downed me man. Sit down here, I wants to tell youse something."

Black Pete reluctantly did as Bill requested and the crowd surged round them to hear what it was this boy from down under had to say to him.

"I takes it you're a bit loaded with licker to-night and perhaps I had the 'vantage of youse for I never lets any of that hootch stuff interfere with me *phys-e-que*, see? Now you think you're some scrapper don't you? Well maybe you are, and I'll give you a fair chanst. Tomorrer youse keep away from the bug-juice, see? and come 'round in de evenin' and I'll spar' a few rounds with youse—tree rounds ull be about enough—just a friendly bout for the sport it will give these gents here. Marquis Queensbury rules or sluggers rules, I don't care which. Youse can go now," and Black Pete promptly sneaked off wishing that an earthquake would open a gulch through Circle and swallow up him, Bill, Jack and everybody else, but it didn't.

All the next day Black Pete wondered how he could get out of the 'friendly bout' that Bill was so willing to *pull off* for the mere fun of the thing. He didn't know what the Marquis of Queensbury rules were but he finally came to the conclusion that he was a better man than his opponent and that the only way he could retrieve his standing in Circle was to give the *Keed* the beating of his life.

Curiously enough he did 'cut out the booze' just as though he had paid Bill for the advice and then he proceeded to get into his best fighting trim.

"I knock heem face een eef I ever heet heem," he said talking to himself, and then to prove to his own satisfaction that he could do it he made four well defined dents in the pine board wall with a smashing blow of his fist.

"An' you said these folks up here was all of the peace-lovin' garden variety, and never use a gun," Bill said soberly when they were in their room after the fracas.

"I thought they were," replied Jack.

"You *thought* they were?" and Bill looked at him as though he had caught him breaking the n^{th} commandment. "Well don't youse think again, Buddy, or youse might hurt yourself, see?"