Mahmudul Hasan

Computing

Quick Submit

Quick Submit

Textile Engineering

Document Details

Submission ID

trn:oid:::1:3124483883

Submission Date

Jan 5, 2025, 4:22 PM GMT+6

Download Date

Jan 5, 2025, 4:23 PM GMT+6

File Name

computing.docx

File Size

14.9 KB

2 Pages

613 Words

3,740 Characters

26% detected as AI

The percentage indicates the combined amount of likely AI-generated text as well as likely AI-generated text that was also likely AI-paraphrased.

Caution: Review required.

It is essential to understand the limitations of AI detection before making decisions about a student's work. We encourage you to learn more about Turnitin's AI detection capabilities before using the tool.

Detection Groups

1 AI-generated only 0%
Likely AI-generated text from a large-language model.

2 AI-generated text that was AI-paraphrased 26%
Likely AI-generated text that was likely revised using an AI-paraphrase tool or word spinner.

Disclaimer
Our AI writing assessment is designed to help educators identify text that might be prepared by a generative AI tool. Our AI writing assessment may not always be accurate (it may misidentify writing that is likely AI generated as AI generated and AI paraphrased or likely AI generated and AI paraphrased writing as only AI generated) so it should not be used as the sole basis for adverse actions against a student. It takes further scrutiny and human judgment in conjunction with an organization's application of its specific academic policies to determine whether any academic misconduct has occurred.

Frequently Asked Questions

How should I interpret Turnitin's AI writing percentage and false positives?
The percentage shown in the AI writing report is the amount of qualifying text within the submission that Turnitin's AI writing detection model determines was either likely AI-generated text from a large-language model or likely AI-generated text that was likely revised using an AI-paraphrase tool or word spinner.

False positives (incorrectly flagging human-written text as AI-generated) are a possibility in AI models.

AI detection scores under 20%, which we do not surface in new reports, have a higher likelihood of false positives. To reduce the likelihood of misinterpretation, no score or highlights are attributed and are indicated with an asterisk in the report (*%).

The AI writing percentage should not be the sole basis to determine whether misconduct has occurred. The reviewer/instructor should use the percentage as a means to start a formative conversation with their student and/or use it to examine the submitted assignment in accordance with their school's policies.

What does 'qualifying text' mean?
Our model only processes qualifying text in the form of long-form writing. Long-form writing means individual sentences contained in paragraphs that make up a longer piece of written work, such as an essay, a dissertation, or an article, etc. Qualifying text that has been determined to be likely AI-generated will be highlighted in cyan in the submission, and likely AI-generated and then likely AI-paraphrased will be highlighted purple.

Non-qualifying text, such as bullet points, annotated bibliographies, etc., will not be processed and can create disparity between the submission highlights and the percentage shown.

High-Performance Computing

1. Modern ISAs and the Evolution of Instruction Sets (RISC vs. CISC) Since the beginning of computing, instruction sets have seen tremendous change. Simple instruction sets that were frequently tailored for the particular hardware they operated on were used by early processors. However, increasingly sophisticated instruction sets were created as computing demands rose.

Reduced Instruction Set Computer, or RISC: A short, highly efficient set of instructions that run in a single clock cycle is the main focus of RISC architectures. MIPS and ARM are two examples. Pipelining is made easier and faster by the simplicity of RISC instructions.

• CISC (Complex Instruction Set Computer): x86 and other CISC designs include a large collection of instructions, some of which carry out intricate operations. This lowers the quantity of instructions needed for a program, but it frequently makes each instruction more complex, which results in longer execution times.

Both strategies are incorporated into contemporary ISAs. For example, contemporary x86 processors maintain an exterior instruction set that is compatible with CISC while utilizing RISC-like micro-operations within.

2. Pipe lining's Function in Control Unit Architecture The process by which several stages of an instruction—fetch, decode, execute, memory access, and write-back—overlap during execution is known as pipelining. This increases throughput by enabling processors to work on several instructions at once.

• Effect on Execution Speed: Pipelining reduces processor component idle time by segmenting tasks into smaller steps and carrying them out concurrently. A five-stage pipeline, for instance, can process five instructions at once, using resources almost continuously.

• Risks: Although pipelining has benefits, it also has risks, including control, structural, and data dangers. To lessen these problems, strategies like branch prediction and forwarding are used.

3. Comparing the Performance of Modern and Older Architectures

• Older Architectures: Pipelining and parallelism were absent from early CPUs, such as the Intel 8086, which ran at slower clock speeds. For instance, it could take thousands of cycles every iteration to run a loop that sums an array.

• current Architectures: Pipelining, SIMD (Single Instruction, Multiple Data), and multi-core designs allow a current ARM Cortex or Intel Core processor to do the identical operation in a fraction of the cycles.

Summing a 1,000-element array is one example.

• Intel 8086: 50,000 cycles, based on an iteration of 50 cycles.

- ARM Cortex-A78: 1,000 cycles (using SIMD and assuming one cycle per loop).

4. Performance Comparison of ARM and x86 ISA Think of a straightforward algorithm: adding up an array of integers.

- ARM: Designed to use less energy and provide clearer instructions. Their speed of execution makes up for the need for additional instructions.

- x86: Complex operations result in fewer instructions, but overall execution time is increased by higher cycle counts per instruction.

Whereas x86 performs well in tasks demanding backward compatibility and broad computing capabilities, ARM performs better in situations requiring energy efficiency and straightforward computations.

5. Suggested Enhancements to ISAs and Control Unit Design

- Better Designs for Control Units

1. Dynamic Instruction Scheduling: Reduce stalls by improving out-of-order execution capabilities.

2. Advanced Branch Prediction: Use predictors based on machine learning to increase the precision of branching choices.

Features of the Instruction Set:

1. Customizable ISAs: Let programmers include domain-specific instructions (like those for cryptography or artificial intelligence).

2. Greater Parallelism: To manage more SIMD operations per cycle.

Justification: Advanced branch prediction and dynamic scheduling increase throughput by lowering pipeline delays.

- For specific applications, customizable ISAs can yield notable performance improvements without necessitating general-purpose overhead.

- In contemporary applications where parallelism is crucial, such as machine learning and big data processing, wider vector units are necessary.

The suggested processor design would provide improved performance, efficiency, and reliability while keeping costs under control by incorporating these advances.