PART 1

RESEARCH

What is the industry/company name?

Computer Hardware Industry

What are their values and goals?

The goal of workplace writing in the computer hardware industry can range from business memos and emails for internal coordination, to technical documents for engineers. In each case the writing has an intended reader and stylistic approach that can differ between companies.

Professional Writing Genre #1 - (due by class time Mon, Jan. 26)

Genre Name **Technical Documentation**

General Analysis: What are the characteristics of the genre? What are its purposes? Who is the audience? Who are the authors? What is the writing style?

Hardware companies produce an extensive amount of technical documents such as IC datasheets, user guides, design outlines and more. These documents are highly specialized, formal, and detail oriented, as the intended audience is engineers, and developers. They assume the reader has adequate background knowledge, using industry specific terminology, and abbreviations. Since technical documentation is very detailed, long, and extensive they are usually written by entire teams of engineers and developers. The purpose of these documents is to teach others who are not directly involved in the development of a certain hardware how to use or understand the workings of a device.

Industry standard practice is highly structured documentation with sections, checklists, tables, diagrams, and step by step instructions. It avoids ambiguous language, instead opting for precise language with a single interpretation. Intentional use of visual diagrams that help readers understand the material. Heavy use of industry specific terminology such as GPIO, power rails, PCB, and DAC.

Find a Primary Source: Copy and paste an example of the genre. Respond to the analytical questions above. What are the characteristics of the genre? What are its purposes? Who is the audience? Who are the authors? What is the writing style? Speak generally about the genre, but also explain a bit about the primary source itself (i.e. the author, their specific purpose, where they work, etc.)

1.3.3 Chip Power-up and Reset Timing

ESP32’ s CHIP_PU pin can enable the chip when it is high and reset the chip when it is low.

When ESP32 uses a 3.3 V system power supply, the power rails need some time to stabilize before CHIP_PU is pulled up and the chip is enabled. Therefore, CHIP_PU needs to be asserted high after the 3.3 V rails have been brought up.

To reset the chip, keep the reset voltage V_{IL_nRST} in the range of (NA ~ 0.6) V. To avoid reboots caused by external interferences, make the CHIP_PU trace as short as possible.

Figure *ESP32 Power-up and Reset Timing* shows the power-up and reset timing of ESP32.

Table *Description of Timing Parameters for Power-up and Reset* provides the specific timing requirements.

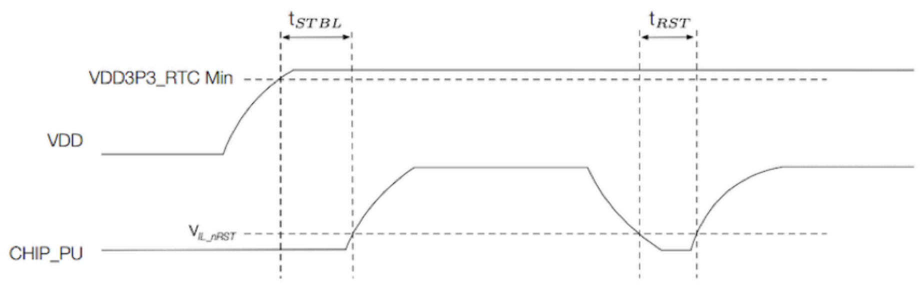


Fig. 4: ESP32 Power-up and Reset Timing

Table 2: Description of Timing Parameters for Power-up and Reset

Parameter	Description	Minimum (µs)
t_{STBL}	Time reserved for the power rails to stabilize before the CHIP_PU pin is pulled high to activate the chip	50
t_{RST}	Time reserved for CHIP_PU to stay below V_{IL_nRST} to reset the chip	50

Attention:

- CHIP_PU must not be left floating.
- To ensure the correct power-up and reset timing, it is advised to add an RC delay circuit at the CHIP_PU pin. The recommended setting for the RC delay circuit is usually R = 10 kΩ and C = 1 µF. However, specific parameters should be adjusted based on the characteristics of the actual power supply and the power-up and reset timing of the chip.
- If the user application has one of the following scenarios:
  - Slow power rise or fall, such as during battery charging.
  - Frequent power on/off operations.
  - Unstable power supply, such as in photovoltaic power generation.

  Then, the RC circuit itself may not meet the timing requirements, resulting in the chip being unable to boot correctly. In this case, additional designs need to be added, such as:
  - Adding an external reset chip or a watchdog chip, typically with a threshold of around 3.0 V.
  - Implementing reset functionality through a button or the main controller.

https://docs.espressif.com/projects/esp-hardware-design-guidelines/en/latest/esp32/esp-hardware-design-guidelines-en-master-esp32.pdf

The following is a snippet from the “ESP32 Hardware Design Guidelines” published by Espressif Systems, a semiconductor company that designs the ESP32 system on chip (SoC). These guidelines are intended for hardware and application engineers to correctly integrate the ESP32 chip into their own commercial hardware products. It outlines best practices, constraints, and electrical requirements to meet the minimum performance standard specified by Espressif.

The snippet is specifically section 1.3.3 of the document which highlights some of the characteristics of technical documentation. Technical documentation is highly structured with sections, checklist, tables, diagrams, and explicit instructions. Authors often make use of visual references to help engineers understand the material better. The diagram (Fig 4) is an example of this conveying the same information as the table below it. However, it was created under the assumption that the reader is familiar with waveform analysis.

When writing technical documentation it's also important to acknowledge that readers will rarely read linearly. Readers will often jump between sections to find the specific information they need.

This has some added implication as certain sections may need to reference information from an earlier portion. Since readers don't usually read in a linear fashion, authors must make heavy use of citations to point the reader to where they could find the information: “Figure ESP32 Power-up and Reset Timing shows the power-up and reset timing of ESP32.“ In this case referencing information within the same section. This also has the added benefit of avoiding the repetition of information between sections.

The actual writing of technical documentation makes heavy use of bullet points to convey information. While they do include short paragraph style writing, they typically try to avoid long paragraph blocks. This stems from authors expecting engineers to scan through a section, rather than reading it in its entirety. Technical documents also are highly detailed and benefit from having the information broken up into concise independent bullet points. They are less susceptible to the ambiguity of reading a string of sentences and piecing them together.

Secondary Source (for example, a person or industry expert talking ABOUT what the genre is & how people in the industry use it): Who is the person? What does the person say? Respond to the analytical questions above based on what the person says. Summarize their perspective here and include a link to the source.

https://medium.com/slalom-build/how-to-write-documentation-that-people-will-actually-read-and-use-b26791fc1429

The author of this article is Hal Deranek who works for Slalom, a business and technology consulting firm in Washington DC. He outlines a few critical criteria with the first being to write for your audience.

When writing documentation there must be a balance between assuming your audience is experienced or completely new. The way you get around this is by “writ[ing] additional documentation” and leveraging the highly structured nature of technical documents. Splitting information into discrete sections of information allows users to skip sections to reach the information they need. Good documentation would hand hold new readers through the technical information while allowing experienced users to go at their own pace.

The next step is to have it peer reviewed by someone “who can validate your documentation” before it is released to the public. This allows for constructive criticism that could fill in some of the gaps of necessary and important information for those who are new.

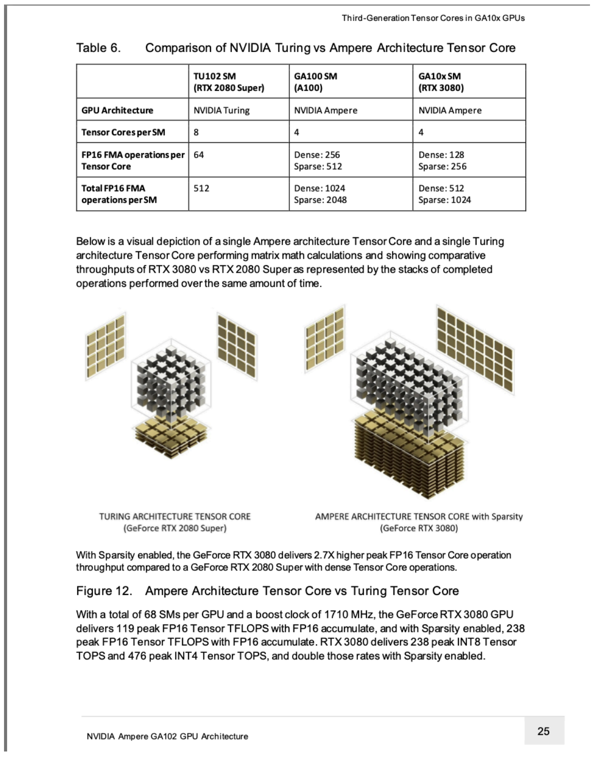
Tertiary Source (for example, a reference guide that explains the genre, including an encyclopedia, Credo, a credible industry database, Claude, Chat GPT and/or Perplexity. You can also use these platforms to find primary and secondary sources that are credible, authentic, and authoritative.) What did you discover? What is the platform and are they credible? What were the prompt/s you used to find this information? And answer the analytical questions above: what did you learn about the characteristics of the genre? What are its purposes? Who is the audience? Who are the authors? What is the writing style?

For the tertiary source, I used ChatGPT’s deep research feature to get a detailed overview of technical documentation as a workplace writing genre. ChatGPT functions as a tertiary source because it makes interpretations of existing first and secondary sources similar to platforms like wikipedia.

ChatGPT draws from a wide range of sources that are publicly available, cited, and credible from major hardware companies such as AMD and NVIDIA. While the tool has the tendency to occasionally hallucinate information or draw the wrong conclusions, it generally produces good and accurate summaries when used with caution. It becomes a powerful tool for building a general understanding of a subject rather than being a factual citation.

The prompt I used provided the genre, industry, and information that I was looking for. This includes purpose, audience, values, writing conventions and more.

What I discovered lined up with my preemptive understanding of the subject highlighting that“ [t]hese documents are **highly formal and detail-rich**, intended for an audience of engineers, developers, and technical stakeholders. They assume a **specialized readership** with background knowledge, using industry-specific terminology and precise language.”



Third-Generation Tensor Cores in GA10x GPUs

Table 6. Comparison of NVIDIA Turing vs Ampere Architecture Tensor Core

	TU102 SM (RTX 2080 Super)	GA100 SM (A100)	GA10x SM (RTX 3080)
GPU Architecture	NVIDIA Turing	NVIDIA Ampere	NVIDIA Ampere
Tensor Cores per SM	8	4	4
FP16 FMA operations per Tensor Core	64	Dense: 256 Sparse: 512	Dense: 128 Sparse: 256
Total FP16 FMA operations per SM	512	Dense: 1024 Sparse: 2048	Dense: 512 Sparse: 1024

Below is a visual depiction of a single Ampere architecture Tensor Core and a single Turing architecture Tensor Core performing matrix math calculations and showing comparative throughputs of RTX 3080 vs RTX 2080 Super as represented by the stacks of completed operations performed over the same amount of time.

TURING ARCHITECTURE TENSOR CORE (GeForce RTX 2080 Super)

AMPERE ARCHITECTURE TENSOR CORE with Sparsity (GeForce RTX 3080)

With Sparsity enabled, the GeForce RTX 3080 delivers 2.7X higher peak FP16 Tensor Core operation throughput compared to a GeForce RTX 2080 Super with dense Tensor Core operations.

Figure 12. Ampere Architecture Tensor Core vs Turing Tensor Core

With a total of 68 SMs per GPU and a boost clock of 1710 MHz, the GeForce RTX 3080 GPU delivers 119 peak FP16 Tensor TFLOPS with FP16 accumulate, and with Sparsity enabled, 238 peak FP16 Tensor TFLOPS with FP16 accumulate. RTX 3080 delivers 238 peak INT8 Tensor TOPS and 476 peak INT4 Tensor TOPS, and double those rates with Sparsity enabled.

NVIDIA Ampere GA102 GPU Architecture 25

ChatGPT used NVIDIA’s GPU architecture whitepaper as an example, which conveyed complex information in a highly structured, objective, and detail oriented manner. The document clearly defines their audience, and purpose while using diagrams, tables, and clear section headings that allow readers to skim and find specific information easily.

The paper itself a high level overview of the design and architecture of NVIDIA GPU more coincided with a briefing or presentation paper.

Based on the overview it gave I drew an understanding that technical documentation is formal, technically detailed, structurable, non linear, and skimmable.

https://www.nvidia.com/content/PDF/nvidia-ampere-ga-102-gpu-architecture-whitepaper-v2.pdf#:~:text=The%20family%20of%20new%20NVIDIA%C2%AE,GPU%20Tensor%20Core%20Architecture%20Whitepaper

NOW IT’S YOUR TURN

How would you use this genre? Who is your audience? What would you say? What is your purpose?

As a hardware developer my work is only as useful as my ability to clearly explain it to others. I would use the genre of technical documentation to describe the architecture and design of the circuit I am creating. This documentation would serve as a reference for implementation details, constraints, and design decisions.

Since this is a highly technical document the intended audience would be other hardware engineers who may need to understand, modify, or build upon my design. Because the intended audience is technically knowledgeable, I would be able to make assumptions like familiarity with common terminology, and electronic concepts.

The overall purpose of this documentation would be to communicate how the system works, how it can be implemented or extended, the limitations, and trouble shooting strategies. Good documentation wouldn't have gaps in necessary or important knowledge as it will be used in a number of ways. Writing in a structured, formal, and detailed manner leaves little room for ambiguity.

The document will be organized into discrete sections covering each component of the circuit. Information will be conveyed primarily through short, concise paragraphs, bullet points, tables, and diagrams. Written in a form that accounts for skimming and non linear reading.

PR-5: Stage Pipelined RISC Architecture

Revision 1.0

1 Overview

The PR-5 is a 32-bit pipelined processor implementing the RISC-5 ISA with a 5: stage pipeline:

1. Instruction Fetch (IF)
2. Instruction Decode (ID)
3. Execute (EX)
4. Memory Access (MEM)
5. Write Back (WB)

2 Architectural Features

- 32-bit fixed width instructions.
- 32 general purpose registers (R0-R31).
- Single cycle ALU operations.
- Separate instruction and data memory interfaces.
- Hazard detection and forwarding unit.
- Static branch predication (predict not taken).

3 Pipelined stages

3.1 Instruction Fetch (IF)

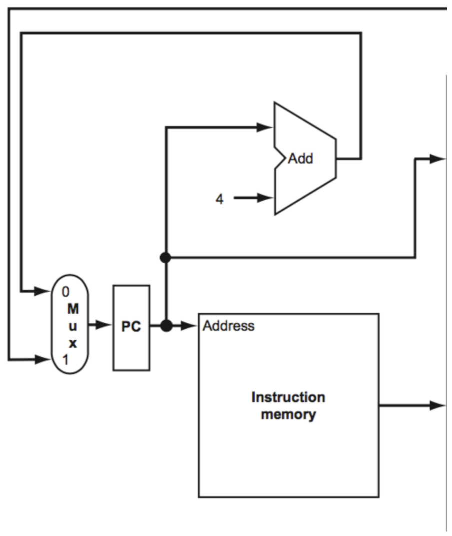
- Program Counter (PC) increments by 4 bytes per cycle.
- Instruction memory is accessed synchronously.
- Branch resolution occurs in the EX stage. If a branch is taken, IF stage is flushed.

Inputs

- PC
- Branch target address (from EX stage)
- Branch taken signal

Outputs

- Instruction
- PC+4



MUX

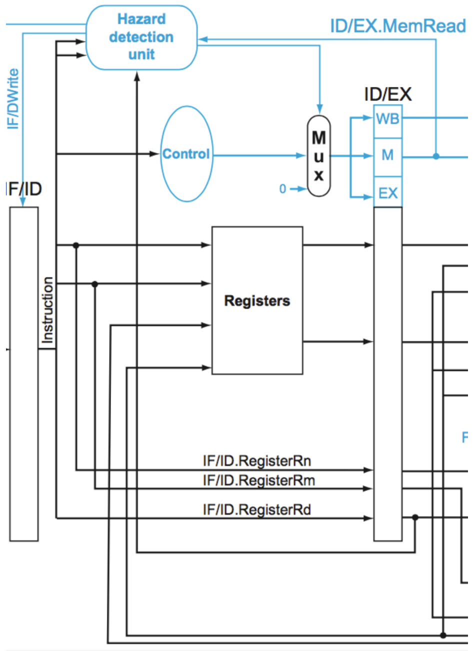
- Input 0: Branch Target address
- Input 1: PC
- Select: Branch taken signal

PC

- Output: Connected to instruction memory
- Input: current PC+4

3.2 Instruction Decode (ID)

- Instruction fields parsed in opcodes, rs1, rs2, rd, and immediate.
- Register file read occurs at this stage.
- Control signals generated.

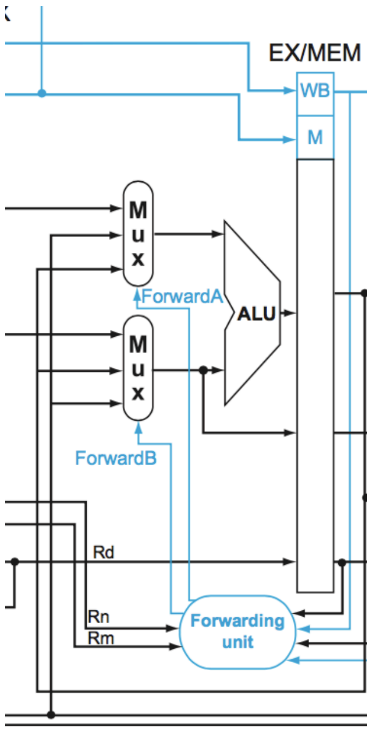


Key Components

- Register File (2 Read P)
- Control Unit
- Hazard Detection Unit

3.3 Execute (EX)

- ALU performs arithmetic and logical operations.
- Branch conditions evaluated.
- Forwarding logic resolves data hazards (from MEM and WB).



ALU Operations:

- ADD, SUB
- AND, OR, XOR
- Shift Left/Shift Right
- Set Less Than

3.4 Memory Access (MEM)

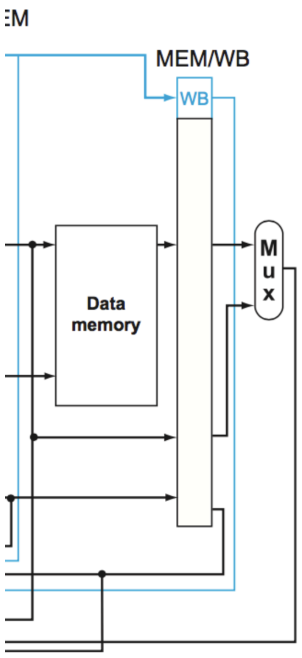
- Load and store instructions access data memory.
- Memory assumes single cycle latency.

Load Behavior

- Data returned at the end of MEM stage.
- Forwarding to EX if required.

3.5 Write Back

- ALU results or memory data written to register file.
- Write occurs on a rising clock edge.



Notes

Mux select signal determines if the ALU result or data memory result is written to the register file.