2025

CS 1105

Digital Electronics & Computer Architecture

ASSIGNMENT ACTIVITY UNIT 6

SANA UR REHMAN

INSTRUCTOR: ROBERT MURRAY

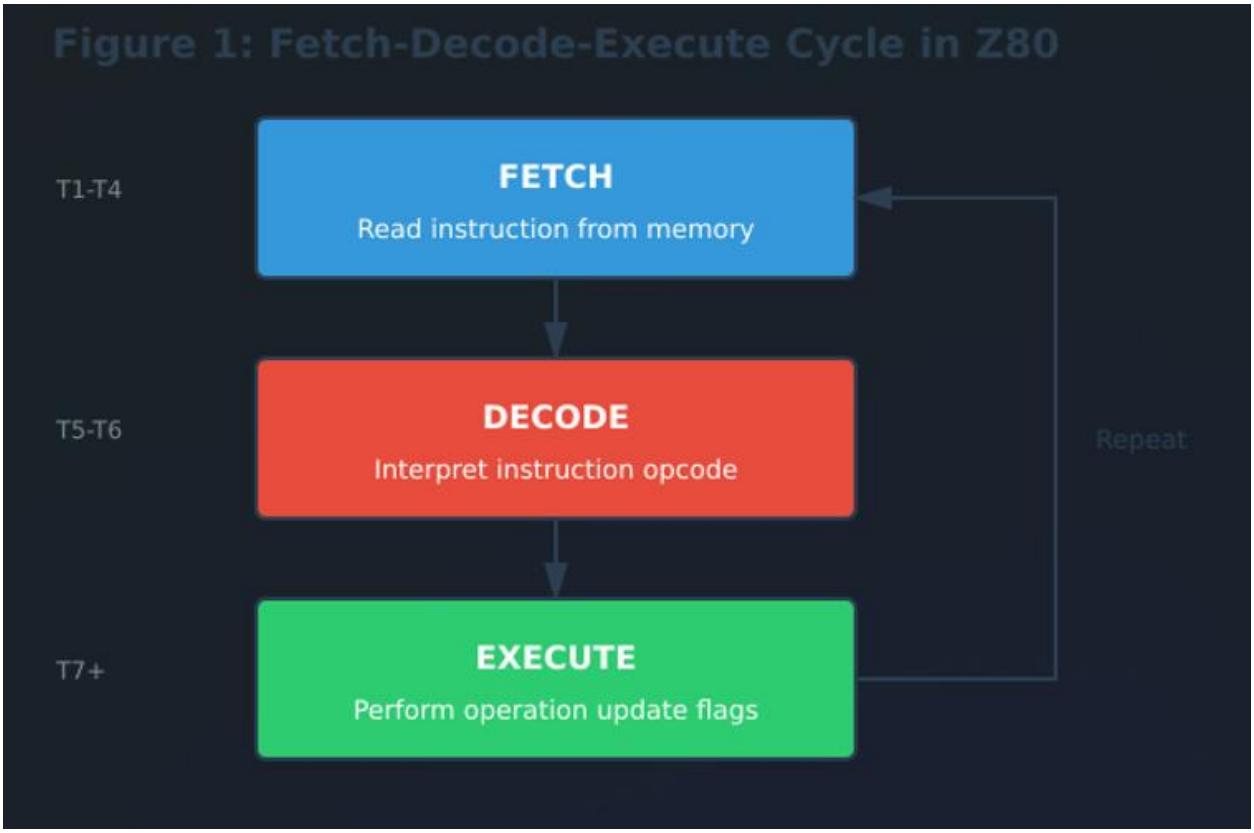
UNDERSTANDING THE Z80 MICROPROCESSOR: INSTRUCTION EXECUTION AND INTERRUPT HANDLING

INTRODUCTION

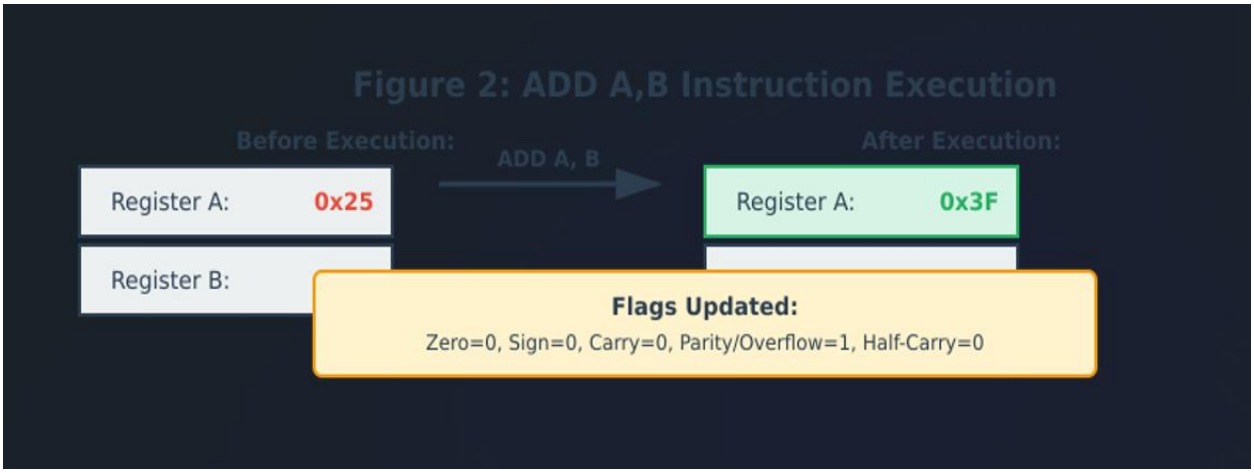
The Z80 microprocessor, developed by Zilog in the mid-1970s, played a vital role in the evolution of microprocessor design. It is an 8-bit processor built on the architecture of the Intel 8080 but enhanced with additional registers, instructions, and interrupting capabilities. Understanding how the Z80 executes instructions and manages interrupts provides insight into fundamental computer architecture concepts such as the fetch–decode–execute cycle, control logic, and real-time responsiveness. These mechanisms remain foundational for modern digital systems and embedded devices (Gaonkar, 2000).

INSTRUCTION EXECUTION IN THE Z80 MICROPROCESSOR

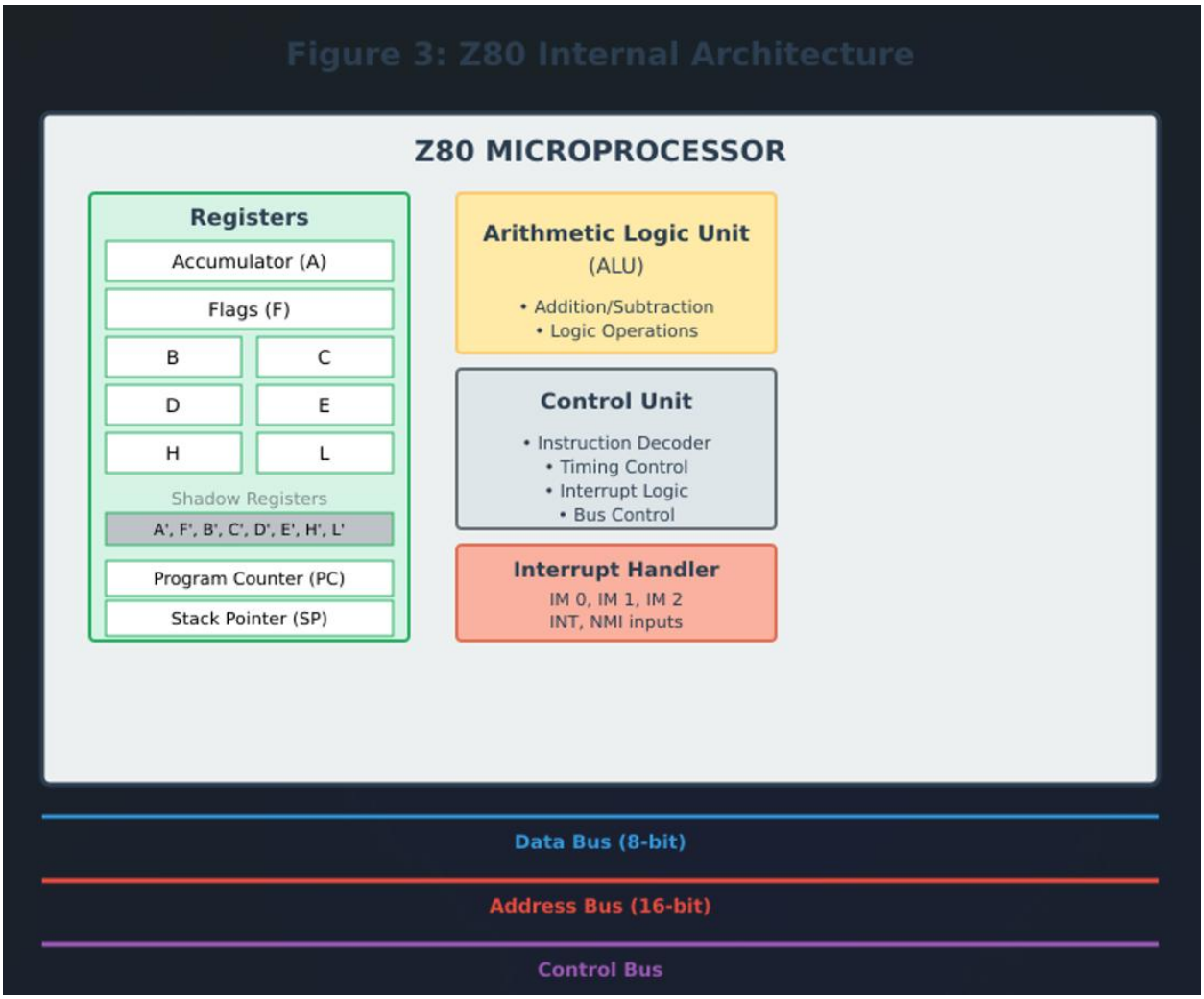
The Z80 microprocessor follows a **fetch–decode–execute** cycle to process instructions sequentially. Each instruction consists of one or more machine cycles, and each cycle contains multiple clock periods known as T-states. During the **fetch cycle**, the microprocessor places the address of the next instruction stored in memory onto the address bus. The memory unit responds by placing the corresponding opcode on the data bus. The Z80 then decodes the opcode internally and generates control signals to execute the operation.



For instance, consider the instruction ADD A, B. During execution, the Z80 decodes the opcode, retrieves the operand from register B, and performs an arithmetic addition with the contents of the accumulator A. The result replaces the value in A, and the relevant flag bits (zero, sign, carry, parity, and half-carry) are updated accordingly.



The Z80's internal architecture includes six general-purpose 8-bit registers (B, C, D, E, H, L) and the accumulator A. It also maintains shadow registers, a stack pointer, and a program counter, which enhances its efficiency and enables rapid context switching. The **instruction decoder** and **arithmetic logic unit (ALU)** collaborate to execute operations such as arithmetic computation, data transfer, and logical manipulation.



Zilog designed the Z80 to support **machine-level pipelining**, where fetching the next instruction overlaps with executing the current one. This overlapping mechanism improves

throughput and minimizes idle clock cycles. As a result, although the Z80 runs at modest clock speeds by modern standards, it demonstrates impressive efficiency for its era (Ganssle, 2008).

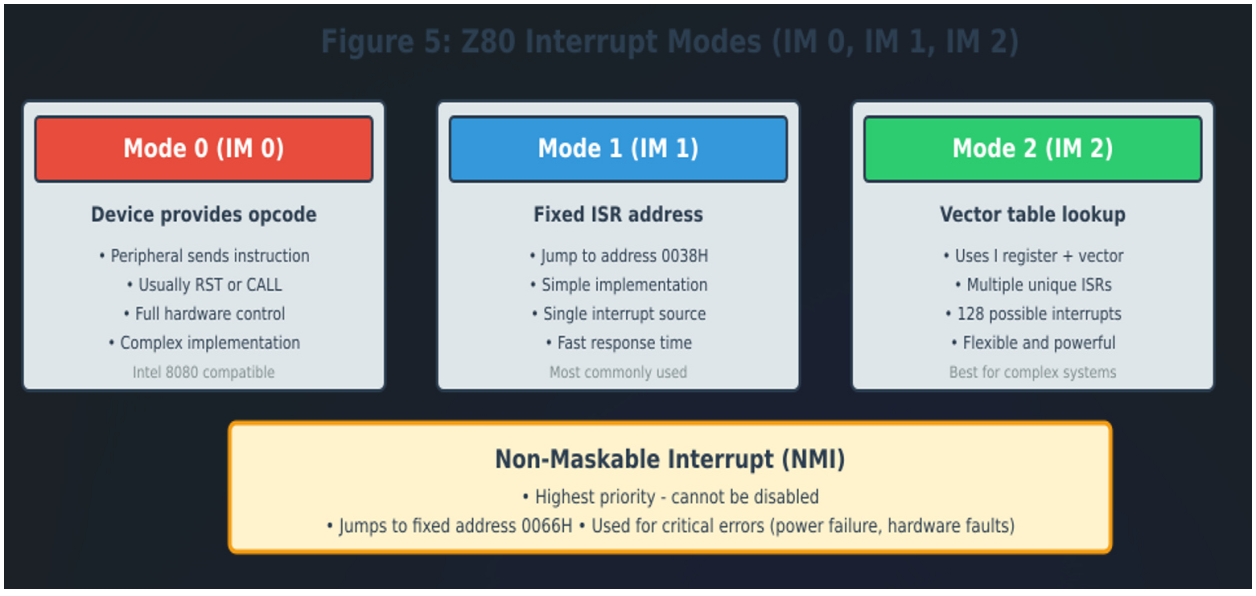
Interrupt Handling in the Z80 Microprocessor

Interrupts enable a microprocessor to respond immediately to internal or external events, ensuring the system operates efficiently without constant polling. The Z80 supports both **maskable** and **non-maskable interrupts (NMI)**, which provide flexibility and reliability in real-time applications.

When an interrupt request (INT) signal is received, the Z80 completes the current instruction before acknowledging the interrupt. It then saves the address of the next instruction onto the stack, allowing the system to resume normal operation once the interrupted service routine (ISR) concludes. The Z80 supports three interrupt modes (IM 0, IM 1, and IM 2), which determine how the processor identifies the address of the ISR:

- **Mode 0:** The external device provides the opcode for the ISR, giving full control to peripheral hardware.
- **Mode 1:** The microprocessor automatically jumps to a fixed ISR address (0038H).
- **Mode 2:** The processor uses a vector table stored in memory, allowing multiple devices to trigger unique interrupt routines.

This structured interrupt mechanism ensures predictable and efficient event handling, crucial in systems like industrial controllers or embedded interfaces. The **non-maskable interrupt (NMI)** has the highest priority and cannot be disabled, which is ideal for critical faults such as hardware failures or power issues.

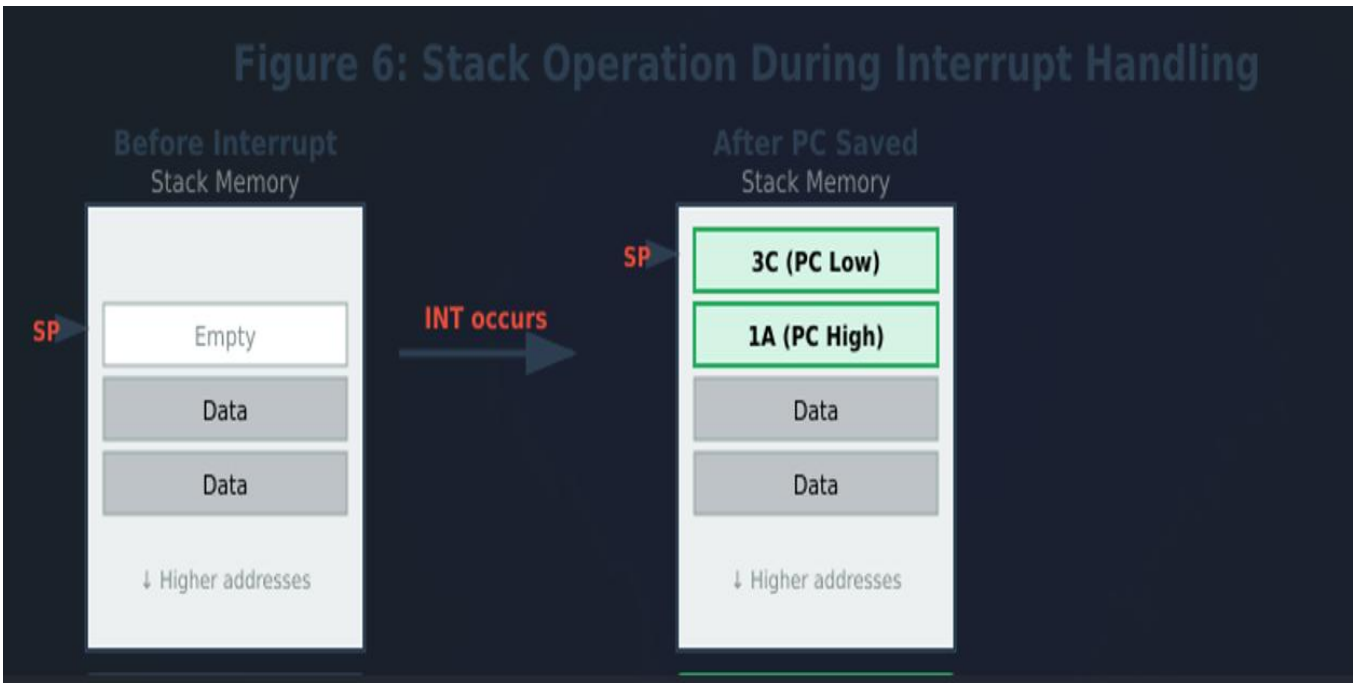


Significance of Interrupt Handling in Ensuring Smooth Operation

Interrupt handling is central to the Z80's reliability and responsiveness. It allows the processor to temporarily halt its main program, service urgent tasks, and then resume execution seamlessly. This process reduces latency, optimizes processing power, and supports multitasking environments.

For example, in an embedded control system, an interrupt could trigger when a sensor detects a threshold condition. The Z80 immediately responds to the input, executes the ISR, and then returns to its main control program. Without interrupts, the processor would need to constantly poll the sensor, wasting cycles and energy.

Furthermore, the Z80's interrupt hierarchy prevents conflicts between simultaneous requests, maintaining deterministic performance. This mechanism laid the groundwork for modern interrupt-driven microcontroller architectures, where real-time event responsiveness is critical for automation and embedded computing.



CONCLUSION

The Z80 microprocessor exemplifies a well-designed balance between simplicity and functionality. Its instruction execution process—rooted in the fetch–decode–execute cycle—illustrates the structured flow of program control, while its interrupt handling mechanisms demonstrate effective real-time responsiveness. By incorporating multiple interrupt modes and prioritized responses, the Z80 ensured stable operation even under complex workloads. Understanding these mechanisms not only strengthens comprehension of digital electronics but also provides a foundation for analyzing modern CPU and microcontroller designs.

REFERENCES

Ganssle, J. (2008). *The Art of Designing Embedded Systems*. Newnes. https://library.uoh.edu.iq/admin/ebooks/46911-ganssle---the-art-of-designing-embedded-systems.pdf

Gaonkar, R. S. (2000). The Z80 microprocessor: Architecture, interfacing, programming, and design. Pearson College Div.

Word Count: 758