Assignment Activity Unit 5

Department of Computer Science, UoPeople

CS 1105-01 - AY2025-T1

Instructor Chinu Singla

**Friday, 18th  October 2024**

### Digital Electronics & Computer Architecture Assignment: Z80 Microprocessor

#### 1. Instruction Execution and Interrupt Handling in the Z80 Microprocessor

The Z80 microprocessor, introduced by Zilog in the 1970s, became a cornerstone of computing due to its efficient instruction execution and interrupt-handling capabilities. The microprocessor has an 8-bit architecture and a 16-bit address bus, allowing it to address 64 KB of memory. Let’s delve into how it executes instructions and manages interrupts.

**Instruction Execution:** The Z80 executes instructions in several steps known as the instruction cycle, comprising four main stages: **fetch**, **decode**, **execute**, and **write back**.

* During the fetch phase, the program counter (PC) points to the memory address of the next instruction, which is then loaded into the instruction register.
* In the decode phase, the control unit interprets the fetched instruction.
* During the execute phase, the instruction is processed by the arithmetic logic unit (ALU), registers, and memory.
* The final phase, write back, updates the necessary registers or memory locations.

This cycle repeats continuously, allowing the microprocessor to process multiple instructions rapidly. The Z80 also has a robust set of instructions, including 158 basic and several extended instructions that manipulate data, control program flow, and interface with peripheral devices.

**Interrupt Handling:** Interrupts are essential in the Z80 for ensuring efficient multitasking. Interrupts are signals sent to the processor to indicate an event that needs immediate attention, temporarily halting the current program to execute a higher-priority task. The Z80 supports multiple interrupt modes, including:

1. **Mode 0**: This mode allows external devices to send an 8-bit instruction that the processor executes.
2. **Mode 1**: A fixed 38H vector address is used when an interrupt occurs.
3. **Mode 2**: This mode allows for a more flexible interrupt vector, where the high byte of the address is supplied by the I register, and the low byte is provided by the device.

The interrupt acknowledgment process in the Z80 involves saving the current state of the processor (registers and program counter) on the stack, executing the interrupt service routine (ISR), and finally returning control to the interrupted program using the **RETI** (Return from Interrupt) instruction. This ensures that the processor can handle the interrupt efficiently and resume normal operations without losing track of the initial program.

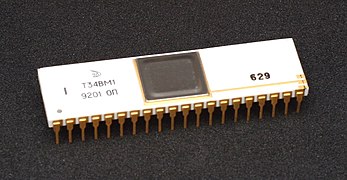


Figure 1: Z80 Microprocessor execution flow (Wikipedia, 2024).

#### 2. Significance of Interrupt Handling in the Z80 Microprocessor

Interrupt handling in the Z80 microprocessor is vital for ensuring the smooth and efficient operation of the system. Interrupts allow the processor to respond to real-time events, such as input from peripheral devices, without the need for continuous polling. This reduces CPU idle time and increases overall system efficiency.

**Significance of Interrupts:**

* **Multitasking**: Interrupts allow the Z80 to pause the current task and handle a higher-priority event, such as an I/O request. This multitasking ability is critical for real-time systems like embedded applications or operating systems.
* **Efficiency**: By using interrupts, the Z80 avoids wasting CPU cycles on polling devices to check their status, thus maximizing the use of processing power.
* **System Responsiveness**: The Z80’s flexible interrupt modes enable it to quickly respond to various devices. For instance, in a peripheral-heavy environment, interrupts allow the processor to react in real-time without delaying ongoing tasks.

**Ensuring Smooth Operation:** To maintain smooth operation during an interrupt, the Z80 uses an interrupt service routine (ISR) to address the interrupt source. The processor halts the main program, services the interrupt, and returns to the exact point where it was interrupted using the RETI instruction. This ability to save and restore the processor’s state ensures that no information is lost during interrupt processing, allowing for seamless program execution.

The Z80 also supports **maskable** and **non-maskable interrupts** (NMI). While maskable interrupts can be disabled when needed, non-maskable interrupts are high-priority and cannot be ignored, ensuring the processor responds to critical situations like system errors.

In conclusion, the Z80’s efficient execution of instructions and its flexible interrupt-handling system make it an exemplary microprocessor for real-time and multitasking environments. Its interrupt modes allow the processor to handle both time-critical tasks and regular operations seamlessly, ensuring both performance and reliability.

**References**

Wikipedia. (2024). Z80 microprocessor. Retrieved from <https://en.wikipedia.org/wiki/Zilog_Z80>