Final Assessment | Fall 2022

Md. Shafayet Hossain

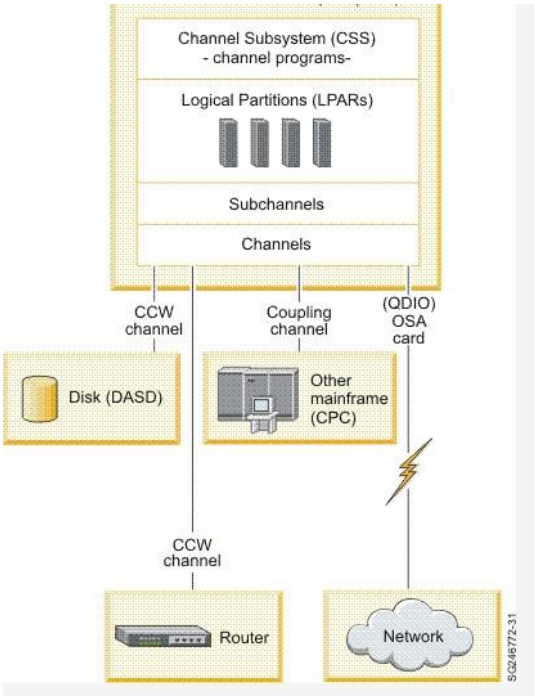
CSE- 21st Batch

Computer Architecture

Code: CSE- 313 | **ID:** 2121210071

Answer to the Question no. 1 (a)

IBM zEC12 I/O Channel Subsystem Structure:



The heart of moving data into and out of a mainframe host is the channel subsystem, or CSS. The CSS is, from a central processor standpoint, independent of the processors of the mainframe host itself. This means that input/output (I/O) within a mainframe host can be done asynchronously.

When an I/O operation is required, the CSS is passed the request from the main processor. While awaiting completion of an I/O request, the main processor is able to continue processing other work. This is a critical requirement in a system designed to handle massive numbers of concurrent transactions. All LPARs within the central processor complex can make use of the channel subsystem.

A simplified example of how the channel subsystem functionally resides within a central processor complex is shown in Figure 1. In this diagram, the large box represents an entire mainframe processor (CPC).

The asynchronous I/O is handled within the channel subsystem by a channel program. Each LPAR ultimately communicates using a sub-channel. In addition, the channel subsystem can be used to communicate between LPARs.

Each CPC has a channel subsystem. Its role is to control communication of internal and external channels to control units and devices.

The channels permit transfer of data between main storage and I/O devices or other servers under the control of a channel program. Some of the other components in Figure 1 are described as follows:

Logical Partition

Within the central processor complex (CPC) are logical partitions that divide the CPC into independent machines that can run any mainframe architecture system control program (for example, z/OS, Linux, or z/VM). Partitions have access to CPC memory and sub-channels.

Sub-channel

The sub-channel represents an I/O device. This is the mechanism by which an I/O request is passed (identified) to the channel subsystem itself. Channel

The channel, represented by a channel path ID or CHPID, represents the actual communication path. A CHPID is the handle by which communication between the CPC and an external device is facilitated.

A CHPID must be unique, since it denotes a unique path of communication for the CPC. The maximum number of allowable CHPIDs within a channel subsystem is 256. Channels can be shared between LPARs.

Historically, a CHPID had a correspondence with a real physical channel connected to the CPC. However, for performance and enhanced capabilities, a CHPID now maps to a physical CHPID (PCHID) using a simple mapping table and a CHPID mapping tool, or CMT.

Control units

One of the main tasks of the channel subsystem is to communicate with storage devices such as tape and direct access storage devices (DASD). This is facilitated by a control unit (which is not shown in Figure 1). Although this is a significant aspect of the channel subsystem, this will not be discussed within this information since it is not a network device.

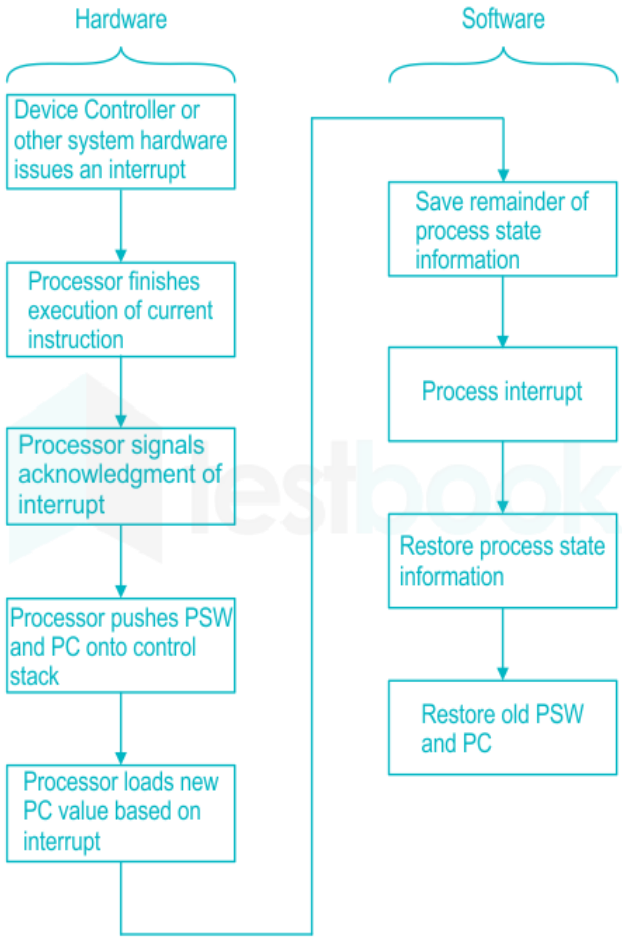
Logical channel subsystem (LCSS)

To facilitate the usage of more CHPIDs, the mainframe architecture supports a logical channel subsystem, or LCSS. The LCSS is functionally identical to the channel subsystem, but up to four LCSSS can be defined within a central processor complex. CHPIDs are unique within the LCSS only; consequently, the 256 CHPID limitation can be overcome.

Answer to the Question no. 1 (b)

When an I/o device completes an I/o operation, the following sequence of hardware events occurs:

1. The device issues an interrupt signal to the processor.
2. The processor finishes the execution of the current instruction before responding to the interrupt.
3. The processor test for an interrupt and sends an acknowledgment signal to the device that issued the interrupt.
4. The processor now needs to save information needed to resume the current program at the point of an interrupt. The minimum information saved to be is the status of the processor and location of the next instruction to be executed.
5. The processor now loads the program counter with the entry location of interrupt handling program.
6. The interrupt handler next processes the interrupt.
7. After interrupt processing, saved register values are retrieved from stack and restored and next instruction will be executed.



Answer to the Question no. 1 (c)

Input is the unit where the processes are initialized. It takes the inputs from the users through the input devices such as keyboard, mouse, scanners and joystick etc.

Data is stored and processed in computers in the form of binary language. So the instructions from the inputs devices are converted into binary languages in the CPU for further process.

Input devices are just a way of communicating with the computer. This is the way through which real world problems are made available to the computer for processing.

Input devices are also responsible for transmitting the data to the main memory of the computer.

Three techniques for inputting a block diagram are:

1. **Drag and drop:** This technique involves selecting pre-made blocks from a library and dragging them onto the diagram canvas. The blocks can then be connected and configured to represent the desired system.
2. **Text-based input:** This technique involves entering text commands to create and connect blocks in the diagram. The syntax and structure of the commands may vary depending on the software being used.
3. **Direct manipulation:** This technique involves directly manipulating blocks and connections in the diagram canvas with a pointing device, such as a mouse. This method allows for quick and intuitive adjustments to the diagram, but may not be as precise or flexible as the other two techniques.

Answer to the Question no. 2 (a)

CACHE MEMORY

Cache memory owes its introduction to Wilkes back in 1965. At that time, Wilkes distinguished between two types of main memory: The conventional and the slave memory. In Wilkes terminology, a slave memory is a second level of unconventional high-speed memory, which nowadays corresponds to what is called cache memory (the term cache means a safe place for hiding or storing things). The idea behind using a cache as the first level of the memory hierarchy is to keep the information expected to be used more frequently by the CPU in the cache TABLE 6.1 Memory Hierarchy Parameters Access type Capacity Latency Bandwidth Cost/MB CPU registers Random 64 – 1024 bytes 1 – 10 ns System clock rate High Cache memory Random 8 – 512 KB 15 – 20 ns 10 – 20 MB/s $500 Main memory Random 16 – 512 MB 30 – 50 ns 1 – 2 MB/s $20 – 50 Disk memory Direct 1 – 20 GB 10 – 30 ms 1 – 2 MB/s $0.25 Tape memory Sequential 1 – 20 TB 30 – 10,000 ms 1 – 2 MB/s $0.025 6.2. CACHE MEMORY 109 (a small high-speed memory that is near the CPU). The end result is that at any given time some active portion of the main memory is duplicated in the cache. Therefore, when the processor makes a request for a memory reference, the request is first sought in the cache. If the request corresponds to an element that is currently residing in the cache, we call that a cache hit. On the other hand, if the request corresponds to an element that is not currently in the cache, we call that a cache-miss.

MAIN MEMORY

Main memory is where programs and data are kept when the processor is actively using them. When programs and data become active, they are copied from secondary memory into main memory where the processor can interact with them. A copy remains in secondary memory.

As the name implies, the main memory provides the main storage for a computer. Two CPU registers are used to interface the CPU to the main memory. These are the memory address register (MAR) and the memory data register (MDR). The MDR is used to hold the data to be stored and/or retrieved in/from the memory location whose address is held in the MAR. It is possible to visualize a typical internal main memory structure as consisting of rows and columns of basic cells. Each cell is capable of storing one bit of information.

Answer to the Question no- 2 (b)

Logical Caches

Cache memory can be located either side of a memory management unit and use either physical or logical addresses as its tag data. In terms of performance, the location of the cache can dramatically affect system performance. With a logical cache, the tag information refers to the logical addresses currently in use by the executing task. If the task is switched out during a context switch, the cache tags are no longer valid and the cache, together with its often hard-won data must be flushed and cleared. The processor must now go to main memory to fetch the first instructions and wait until the second iteration before any benefit is obtained from the cache. However, cache accesses do not need to go through the MMU and do not suffer from any associated delay.

Physical Caches

Physical caches use physical addresses, do not need flush-ing on a context switch and therefore data is preserved within the cache. The disadvantage is that all accesses must go through the memory management unit, thus incurring delays. Particular care must also be exercised when pages are swapped to and from disk.

If the processor does not invalidate any associated cache entries, the cache contents will be different from the main memory con-tents by virtue of the new page that has been swapped in.

Of the two systems, physical caches are more efficient, providing the cache coherency problem is solved and MMU delays are kept to a minimum. RISC architectures like the PowerPC solve the MMU delay issue by coupling the MMU with the cache system. An MMU translation is performed in conjunction with the cache look up so that the translation delay overlaps the memory access and is reduced to zero. This system combines the speed advantages of a logical cache with the data efficiency of a physical cache.

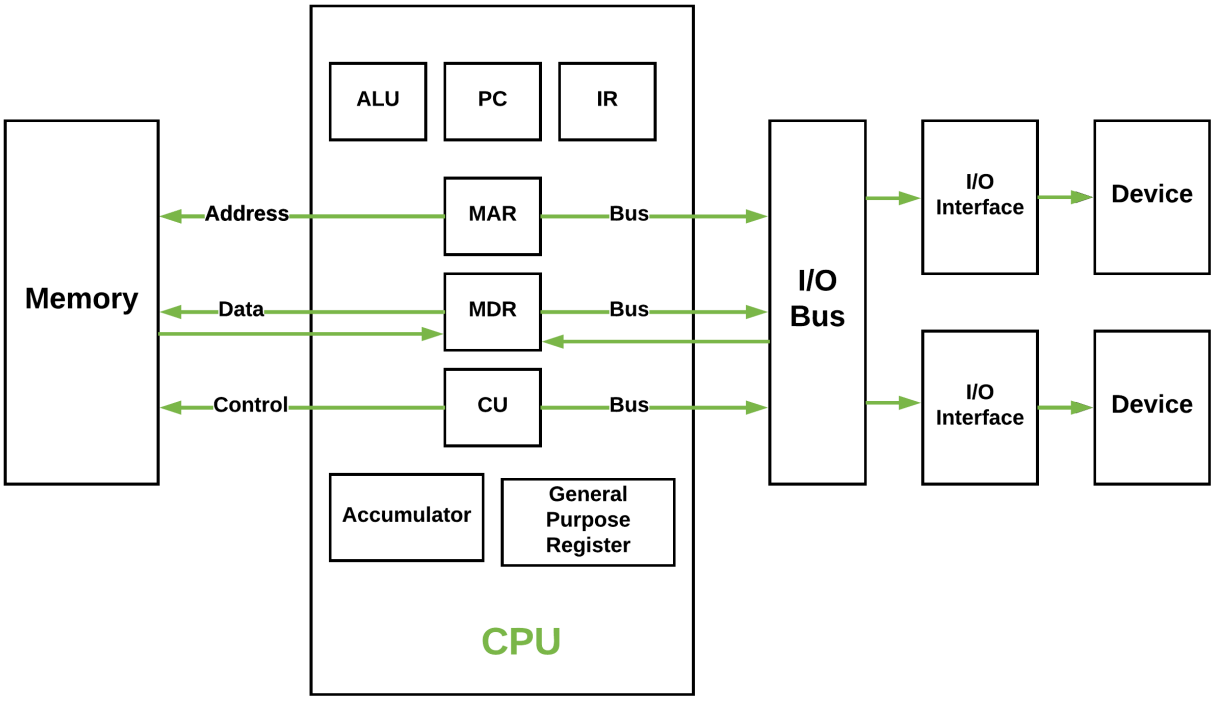
Answer to the Question no- 3 (a)

The main components of a CPU (Central Processing Unit) are:

- Arithmetic Logic Unit (ALU): performs arithmetic and logical operations on data.
- Control Unit (CU): manages the flow of data and commands within the CPU and to other components.
- Register: small, high-speed storage units used to temporarily hold data and instruction.

The CPU interacts with main memory (RAM) to store and retrieve data and instructions. The CPU sends a request to main memory to fetch data or an instruction, and main memory sends it back to the CPU for processing.

The CPU also interacts with I/O (input/output) devices, such as keyboard, mouse, and display, to receive or send data. The CPU sends a request to an I/O device to perform an action, such as receiving user input, and the device sends the result back to the CPU.



Answer to the Question no- 3 (b)

The indirect addressing in x86 family:

Indirect addressing in x86 family refers to a mode of accessing memory in which the effective address is calculated based on the contents of a register or memory location, rather than a constant value. This allows for more flexibility in accessing memory, as the effective address can be dynamically changed during program execution. The syntax for indirect addressing in x86 assembly language uses square brackets to indicate that the contents of the specified register or memory location should be used as an offset to determine the effective address. For example, the instruction **mov eax, [ebx]** would move the contents of the memory location pointed to by the value stored in the **ebx** register into the **eax** register.

Example:

```
MOV AX, [BX]  ; Suppose the register BX contains 4895H, then the contents
              ; 4895H are moved to AX
ADD CX, {BX}
```