Victoria University of Bangladesh
Name: Md. Ziaul Hoque “Sohel”
Student ID: 2221220031
Course Title: Computer Architecture
Course Code: CSE 313
Batch: 22nd(evening)

Ans to the Que No 1(A)

IBM zEC12 I/O Channel Subsystem:

Introduced in August 2012, the zEnterprise EC12 is based on the zEC12 chip, a 5.5 GHz 8-core out-of-order CISC-based zArchitecture processor. The zEC12 can have a maximum of 120 cores, 101 of which are customer configurable to run operating systems and applications.[27] The maximum number of cores available in a particular model of the zEC12 is denoted by the model name. For example, the H20 has up to 20 cores orderable for direct customer use, plus spare and a special I/O processor core type, the System Assist Processor. Each core can be characterized as a Central Processor (CP), Integrated Facility for Linux (IFL) processor, z Application Assist Processor (zAAP), z10 Integrated Information Processor (zIIP), Internal Coupling Facility (ICF) processor, or additional System Assist Processor (SAP). The zEnterprise EC12 allows up to 3 TB (usable) of redundant array of independent memory (RAIM).
The EC12 has 50% higher total capacity than the z196 (up to 78,000 MIPS), and supports Transactional Execution and Flash Express – integrated SSDs which improve paging and certain other I/O performance.

Ans to the Que No 1(B)

When an I/O device completes an I/O operation, the following sequence of hardware events occur:

1. Interrupt Request (IRQ): The I/O device signals the CPU that the operation has completed by sending an interrupt request.
2. Interrupt Service Routine (ISR): The CPU stops its current task and jumps to the interrupt service routine associated with the interrupt request.
3. DMA Transfer Completion: If Direct Memory Access (DMA) was used for the I/O operation, the DMA controller signals completion.
4. CPU Acknowledgment: The CPU acknowledges the completion of the I/O operation by sending an acknowledgment signal to the I/O device.
5. Return from Interrupt: The CPU returns to its previous task after executing the ISR.

This sequence allows the CPU to handle multiple I/O operations concurrently, improving overall system performance.

Ans to the Que No 1(C)

I/O operation deals with the exchanges of data between the memory and the external devices either in the direction to the memory (READ) or in the direction from the memory (WRITE). But the problem arise on how the processor will manage the flow of data to and from the external devices in term of transfer speed, processor idle time, complexity and etc. So in general, there are three technique for I/O operation , which are:

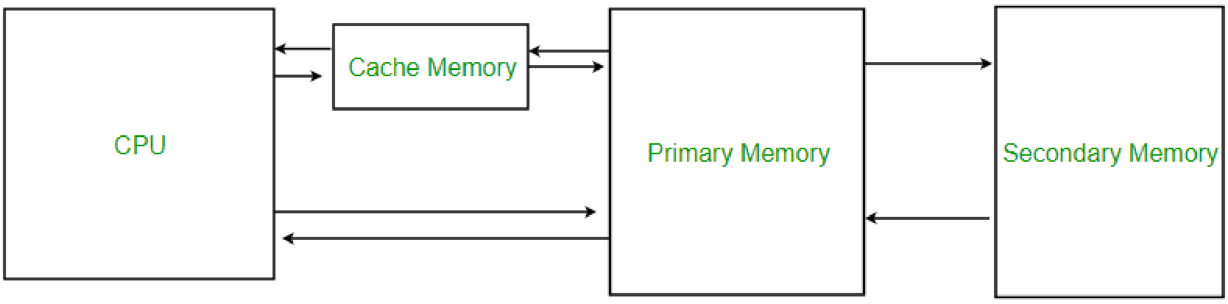
Programmed I/O : Programmable I/O is one of the I/O technique other than the interrupt-driven I/O and direct memory access (DMA). The programmed I/O was the most simple type of I/O technique for the exchanges of data or any types of communication between the processor and the external devices. With programmed I/O, data are exchanged between the processor and the I/O module. The processor executes a program that gives it direct control of the I/O operation, including sensing device status, sending a read or write command, and transferring the data.

Interrupt driven I/O : Interrupt driven I/O is an alternative scheme dealing with I/O. Interrupt I/O is a way of controlling input/output activity whereby a peripheral or terminal that needs to make or receive a data transfer sends a signal. This will cause a program interrupt to be set. At a time appropriate to the priority level of the I/O interrupt. Relative to the total interrupt system, the processors enter an interrupt service routine. The function of the routine will depend upon the system of interrupt levels and priorities that is implemented in the processor.

Direct Memory Access : Differ from Programmed I/O and Interrupt-Driven I/O, Direct Memory Access is a technique for transferring data within main memory and external device without passing it through the CPU. DMA is a way to improve processor activity and I/O transfer rate by taking-over the job of transferring data from processor, and letting the processor to do other tasks. This technique overcomes the drawbacks of other two I/O techniques which are the time consuming process when issuing command for data transfer and tie-up the processor in data transfer while the data processing is neglected. It is more efficient to use DMA method when large volume of data has to be transferred. For DMA to be implemented, processor has to share its' system bus with the DMA module.

Ans to the Que No 2(A)

Cash Memory:



Cache Memory is a special very high-speed memory. It is used to speed up and synchronize with high-speed CPU. Cache memory is costlier than main memory or disk memory but more economical than CPU registers. Cache memory is an extremely fast memory type that acts as a buffer between RAM and the CPU. It holds frequently requested data and instructions so that they are immediately available to the CPU when needed. Cache memory is used to reduce the average time to access data from the Main memory. The cache is a smaller and faster memory that stores copies of the data from frequently used main memory locations. There are various different independent caches in a CPU, which store instructions and data.

Primary Memory (Main Memory)

The main memory is the fundamental storage unit in a computer system. It is associatively large and quick memory and saves programs and information during computer operations. The technology that makes the main memory work is based on semiconductor integrated circuits.
RAM is the main memory. Integrated circuit Random Access Memory (RAM) chips are applicable in two possible operating modes are as follows −

- **Static** – It consists of internal flip-flops, which store the binary information. The stored data remains solid considering power is provided to the unit. The static RAM is simple to use and has smaller read and write cycles.

- **Dynamic** – It saves the binary data in the structure of electric charges that are used to capacitors. The capacitors are made available inside the chip by Metal Oxide Semiconductor (MOS) transistors. The stored value on the capacitors contributes to discharge with time and thus, the capacitors should be regularly recharged through stimulating the dynamic memory.

Ans to the Que No 2(B)

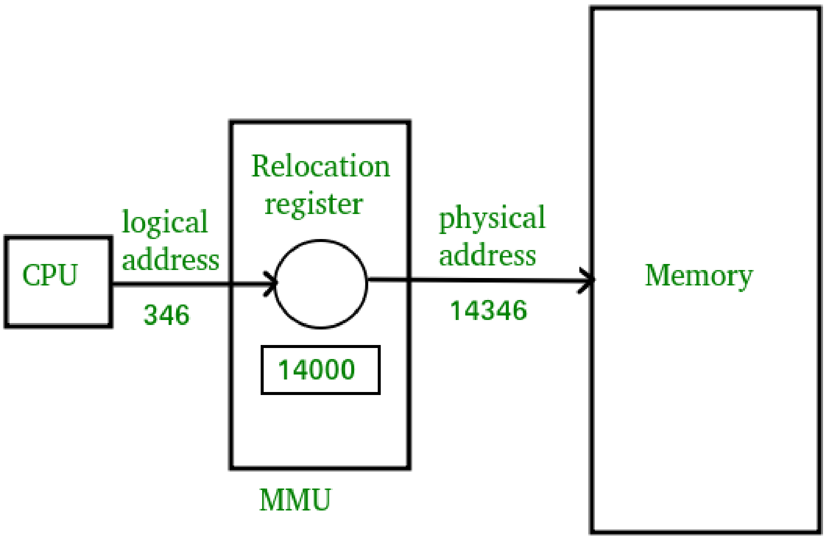
Logical and Physical Caches:

Logical Address:

Logical Address is generated by CPU while a program is running. The logical address is virtual address as it does not exist physically, therefore, it is also known as Virtual Address. This address is used as a reference to access the physical memory location by CPU. The term Logical Address Space is used for the set of all logical addresses generated by a program's perspective.
The hardware device called Memory-Management Unit is used for mapping logical address to its corresponding physical address.

Physical Address:

Physical Address identifies a physical location of required data in a memory. The user never directly deals with the physical address but can access by its corresponding logical address. The user program generates the logical address and thinks that the program is running in this logical address but the program needs physical memory for its execution, therefore, the logical address must be mapped to the physical address by MMU before they are used. The term Physical Address Space is used for all physical addresses corresponding to the logical addresses in a Logical address space.

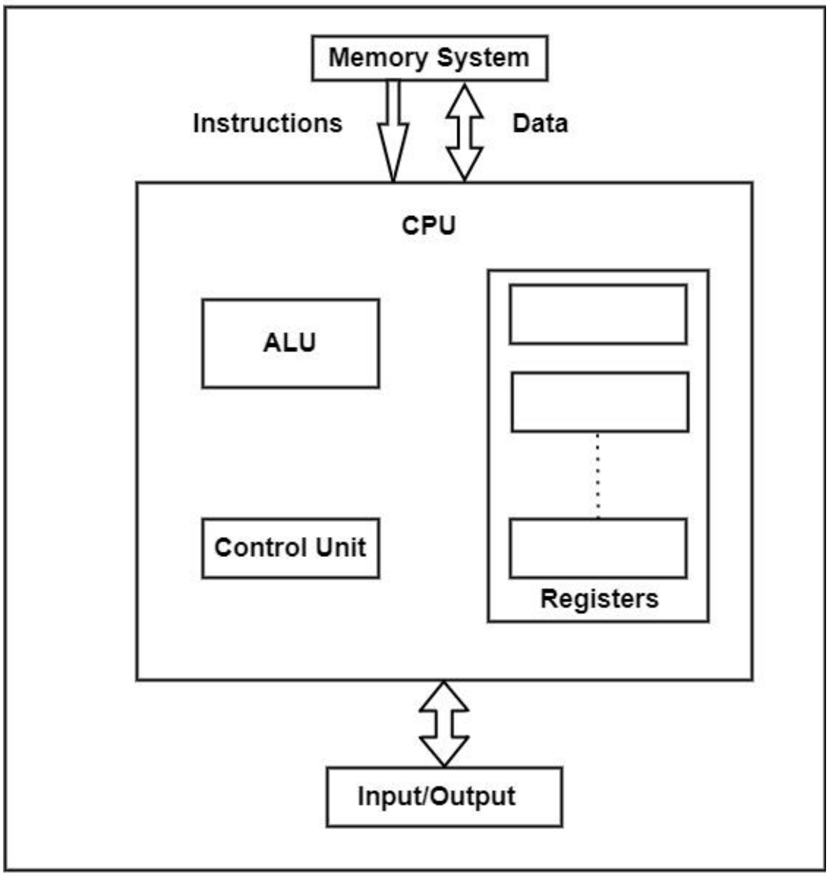


Ans to the Que No 3(A)

Components of CPU:

A CPU includes three major components that are as follows –

Components of CPU



Register Set

The register set contrasts from one system to another. The register set includes several registers which contain general-purpose registers and special-purpose registers. The general-purpose registers do not implement any particular function. They save the temporary information that is needed by a program. The special-purpose registers execute various functions for the CPU.

ALU

The ALU implements all the arithmetic, logical, and shift operations by supporting important circuitry that provides these evaluations.

Control Unit

The control unit fetches the instructions from the main memory, decodes the instructions, and then executes them.

The CPU interacts with the main memory and input/output devices. The CPU reads and writes data to and from the memory system and transfers data to and from the I/O devices.

An elementary execution cycle in the CPU can be defined as follows –
The CPU fetches the instruction to be implemented from the main memory and saves it in the Instruction Register (IR).

The instruction is decoded.
The operands are fetched from the memory system and stored in the CPU registers.
The instructions are then executed.
The results are transferred from the CPU registers to the memory system.
If there are more instructions to be executed, the execution cycle repeats. Some pending interrupts are also tested during the execution cycle.

Example – The interrupts including I/O device request, arithmetic overflow, or pages are tested during the execution cycle.
The procedures of the CPU are represented by the micro-orders issued by the control unit.
The micro-orders are the control signals, which are transfer over-determined control lines.
CPU is the main component of a computer system. It can give out each instruction of a program to implement primary arithmetical, logical, and input/output operations.

Ans to the Que No 3(B)

Register Indirect Addressing (aka *indirect addressing mode*) often used for addressing data arrays inside programming loops:

- Effective address of operand contained in a register.
- For 32-bit addressing, all 32-bit registers can be used.
- For 16-bit addressing, the offset value can be in one of the three registers: BX, SI, or DI:
- mov bx, offset Table ; Load address
- add ax, [bx]   ; Register indirect addressing
- Square brackets [BX] indicate that BX is holding a memory offset.
- Operand [BX] serves as a *pointer* to data in memory.
- Register indirect can be used to implement arrays. For example, to sum an array of word-length integers,

```
  mov cx, size          ; set up size of Table
  mov bx, offset Table  ; BX <- address of Table
  xor ax, ax            ; zero out Sum
Loop1:
  add ax, [bx]
  inc bx     ; each word is 2 bytes long, so
  inc bx     ; need to increment BX twice!
  loop Loop1
```

END