1. [7 points]
   (i) (3 points) With the help of an example code, explain what is meant by *false dependencies* such as output dependencies and anti-dependencies.

(ii) (4 points) Describe two techniques to overcome the impact of false dependencies. Mention at least one drawback of your proposed methods.
2. [7 points]
(i) (2 points) Explain what is wrong with the statement: “An assembly language should provide only a handful of pseudo instructions (e.g., `nop` instruction for MIPS), because providing a large number of pseudo instructions would significantly increase the hardware complexity.”

(ii) (2 points) Explain what is meant by **hardware interrupt** and **software interrupt** and the need to have both types of interrupts.

(iii) (3 points) Explain why many hardware systems include a **Translation Lookaside Buffer (TLB)**. What type of address mapping is typically used in a TLB? Why?
3. [6 points]
(i) (3 points) Give two reasons in favor of and one reason against using a large page size in a virtual memory system.

(3 points)
Describe how an inverted page table is structured and accessed. Explain one advantage and one disadvantage of using inverted page table.
1. [7 points]

(i) (3 points) With the help of an example code, explain what is meant by false dependencies such as output dependencies and anti-dependencies.

False dependencies are dependencies that arise due to reuse of storage locations such as registers and memory locations. Consider the following code segment:

```
1: R1 <-- R2 + R3
2: R2 <-- R3 + R4
3: R4 <-- R2 + R5
4: R1 <-- R6 + R7
```

In this code segment, instruction 2 has a false dependency (anti-dependency) with instruction 1 through register R2. If these two instructions are executed out of order, instruction 1 may get an incorrect value for R2. Similarly, there is a false dependency (output dependency) between instructions 1 and 4 through register R1. If these two instructions are executed out of order, the value retained in R1 after the execution of the code segment would be that produced by instruction 1, and therefore could potentially be incorrect.

(ii) (4 points) Describe two techniques to overcome the impact of false dependencies. Mention at least one drawback of your proposed methods.

- a. One technique to overcome false dependencies is through static renaming (by the compiler). One drawback of static renaming (of registers) is that it could increase the number of memory references, thereby affecting performance.

- b. Another technique to overcome false dependencies is to do dynamic renaming (by the hardware). One drawback of dynamic renaming is the additional hardware complexity that it introduces, resulting in potential increase in cycle time and power consumption.
2. [7 points]

(i) (2 points) Explain what is wrong with the statement: “An assembly language should provide only a handful of pseudo instructions (e.g., nop instruction for MIPS), because providing a large number of pseudo instructions would significantly increase the hardware complexity.”

Pseudo instructions are translated by the assembler into one or more machine-language instructions that are already supported by the instruction set architecture. Because the hardware only implements the instructions that are supported by the instruction set architecture, adding additional pseudo instructions does not increase the hardware complexity (it only increases the assembler complexity, albeit slightly).

(ii) (2 points) Explain what is meant by hardware interrupt and software interrupt and the need to have both types of interrupts.

A hardware interrupt is an interrupt raised (on an interrupt pin) by an I/O device. Such interrupts are asynchronous in nature, and very likely to be unrelated to the process currently running on the processor. Examples are: (i) keyboard interrupt when a key is pressed on the keyboard, and (ii) timer interrupt when a predetermined time period has elapsed. A system needs to support hardware interrupts to incorporate multiprogramming.

A software interrupt is an interrupt raised by the program that is currently being executed on the processor. Such interrupts are synchronous in nature, because they occur at exactly the same place in program execution when the same program is run multiple times with the same set of inputs. A system needs to support software interrupts so that the running process can request services from the operating system. Examples are: (i) read 16 bytes from an open file, and (ii) fork a child process.

(iii) (3 points) Explain why many hardware systems include a Translation Lookaside Buffer (TLB). What type of address mapping is typically used in a TLB? Why?

Most computer systems incorporate a large amount of virtual memory, resulting in large page tables. With the use of large page tables, only a small portion of the page tables can realistically be placed in physical memory; the rest have to be placed in the hard disk. This means that every virtual address-to-physical address translation requires accessing physical memory (or even the hard disk), making memory accesses very slow. In order to speed up the translation, the memory management unit (MMU) maintains a cache memory that keeps a copy of the recent address translations. This special cache memory is called the translation lookaside buffer (TLB).

The TLB typically uses fully associative mapping (or some form of it). This is because a miss in the TLB would necessitate accessing the page table, which can be very time-consuming.
3. [6 points]
(i) (3 points) Give two reasons in favor of and one reason against using a large page size in a virtual memory system.

The use of large page sizes:
(i) results in smaller page tables,
(ii) increases the TLB coverage (the total virtual memory that is covered by all of the entries in the TLB), and
(iii) decreases page faults if there is good spatial locality.

Reasons against using a large page size are:
(i) a page fault takes more time to service, and
(ii) internal fragmentation.

(3 points)
Describe how an inverted page table is structured and accessed. Explain one advantage and one disadvantage of using inverted page table.

An inverted page table stores the physical frame-to-virtual page mapping. It associates one entry for each physical frame, and therefore has as many entries as the number of physical frames. Each entry stores the virtual page number that currently maps to that physical frame, if any. Because the processor generates virtual addresses (from which the MMU generates the virtual page number (VPN)), the inverted page number can be accessed only with the VPN as the key. This means that either a sequential access or an associative lookup is required. The first option has significant time delay whereas the second option involves hardware complexity. One solution to deal with this problem is to use a hash anchor table (that employs a hashing function).

The main advantage of an inverted page table is its small size, and the potential to keep it in physical memory itself, or even within the MMU. One disadvantage of an inverted hash table is the hardware complexity.