The instructions below must be followed strictly. Failure to do so can result in serious grade loss.

⇒ **DO NOT OPEN THIS EXAM UNTIL TOLD TO DO SO.**

⇒ **Read all questions very carefully before answering them.**

⇒ **Check the number of papers in the question sheet and make sure the paper is complete.**

**Specific instructions:**

<table>
<thead>
<tr>
<th>Open book/notes, help sheet:</th>
<th>Closed Book/Closed Notes/ No Help Sheet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Calculator usage:</td>
<td>Desirable</td>
</tr>
<tr>
<td>Write in pen/pencil:</td>
<td>Anything legible, but neatly please</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Any other instruction(s):</th>
<th>Attempt all questions ON this questionnaire. Be concise and answer in accordance to the allocated marks. Try to avoid overwriting/ cutting, etc.</th>
</tr>
</thead>
</table>

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>7</td>
<td>13</td>
<td>19</td>
<td>25</td>
</tr>
<tr>
<td>2</td>
<td>8</td>
<td>14</td>
<td>20</td>
<td>26</td>
</tr>
<tr>
<td>3</td>
<td>9</td>
<td>15</td>
<td>21</td>
<td>27</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td>16</td>
<td>22</td>
<td>28</td>
</tr>
<tr>
<td>5</td>
<td>11</td>
<td>17</td>
<td>23</td>
<td>29</td>
</tr>
<tr>
<td>6</td>
<td>12</td>
<td>18</td>
<td>24</td>
<td></td>
</tr>
</tbody>
</table>
Part A [20 marks]

Please encircle the correct answers. Each question carries 1 mark.

1. Which of the following is a memory that can only hold its charge for a short time and must be refreshed periodically?
   a. DRAM
   b. SRAM
   c. SDRAM
   d. EPROM
   e. EEPROM

2. Which of the following non-volatile memory is electrically erasable and programmable at the block level (multiple bytes at a time)?
   a. EPROM
   b. SRAM
   c. SDRAM
   d. Flash
   e. EEPROM

3. Which of the following ranks highest (smallest, most expensive, fastest access, most frequently accessed) in the memory hierarchy?
   a. hard drive
   b. cache
   c. CD-ROM
   d. RAM
   e. registers

4. Which of the following ranks second in the memory hierarchy?
   a. hard drive
   b. cache
   c. CD-ROM
   d. RAM
   e. registers

5. Which of the following ranks third in the memory hierarchy?
   a. hard drive
   b. cache
   c. CD-ROM
   d. RAM
   e. registers

6. Which of the following ranks the lowest (largest, least expensive, slowest access, least frequently accessed) in the memory hierarchy?
   a. hard drive
   b. L1-cache
   c. L2-cache
   d. RAM
   e. registers
7. The concentric set of rings on a disk platter are called:
   a. gaps
   b. heads
   c. sectors
   d. tracks
   e. worms

8. The conducting coils that read and write data on a disk platter are called:
   a. gaps
   b. heads
   c. sectors
   d. tracks
   e. worms

9. The block-size regions that typically hold about 512 bytes of data on a disk platter called:
   a. gaps
   b. heads
   c. sectors
   d. tracks
   e. worms

10. The areas of separation on a disk platter that break up the areas of physical data storage are called:
    a. gaps
    b. heads
    c. sectors
    d. tracks
    e. worms

11. Which member of the memory hierarchy is regarded as the lowest-cost, slowest speed media?
    a. magnetic disk
    b. CD-ROM
    c. DVD
    d. magnetic tape
    e. Zip Drive

12. This technology, popular for storing digitized video information, has a capacity of over 5 gigabytes:
    a. magnetic disk
    b. CD-ROM
    c. DVD
    d. magnetic tape
    e. Zip Drive

13. The time that it takes the read/write head of a disk drive to position under the desired sector once it is positioned on the right track is called the:
    a. access time
b. rotational delay  
c. seek time  
d. transfer time

14. This time is dependent on how many bytes of data will be transferred from the disk:
   a. access time  
b. rotational delay  
c. seek time  
d. transfer time

15. The time that is required to move the read/write head to the desired track is called the:
   a. access time  
b. rotational delay  
c. seek time  
d. transfer time

16. The time that it takes to move the read/write head to both the desired track and the desired sector is called the:
   a. access time  
b. rotational delay  
c. seek time  
d. transfer time

Please answer briefly in the space provided

17. RAID is an acronym for ________ Redundant Array of Independent Disks ________

18. The three principal I/O techniques are:
   i) ________ Programmed I/O ________
   ii) ________ Interrupts ________
   iii) ________ DMA ________

19. Label the four main structural components of a computer system

   COMPUTER
   ____________________________
   | Input / Output |
   | Main Memory    |
   | System Interconnection |
   | Central Processing Unit |
20. **Briefly distinguish between memory access time and memory cycle time.**

   The memory access time is the time it takes to perform a read or write operation (for RAM), i.e. the time from the instant that an address is presented to the memory to the instant that the data have been stored or made available for use. Memory cycle time consists of the access time plus any additional time before a second access can commence and concerned with the system bus.

---

**Part B**

21. **[15 marks]** Consider a 16-bit hypothetical machine Dino-Processor of the pre-Cambrian era that contains a single data register, the accumulator (AC) with the following characteristics:

<table>
<thead>
<tr>
<th>0</th>
<th>3</th>
<th>4</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>Opcode</td>
<td>Address</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

   (a) Instruction Format

   Dino-processor used 16-bit instructions. Four bits (0-3) were used for the opcode. The remaining twelve bits (4-15) were used for the memory address. Both Instructions and data are 16 bits long.

   Dino-processor's registers included the following for the indicated purpose:

   
<table>
<thead>
<tr>
<th>Program Counter (PC)</th>
<th>Address of next Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instruction Register (IR)</td>
<td>Instruction being executed</td>
</tr>
<tr>
<td>Accumulator (AC)</td>
<td>Temporary storage</td>
</tr>
</tbody>
</table>

   Some of the Dino-processor's opcodes were:

   0000 Load AC from memory
   0001 Store AC to memory
   0010 Add to AC from memory
   0011 Multiply by AC from memory
   0100 Jump statement - Load PC from memory
   0111 Halt the program

   (b) Partial List of Opcodes
The initial state of the memory (in hexadecimal notation) is detailed on page 6. Suppose the program counter is initialized so that the instruction at location 100h is executed first.

Step through the program, updating the appropriate registers and memory locations, until the Halt instruction is reached.

Answer the questions using the table provided on the next page detailing the initial contents of memory. You may also use the reverse side for rough work if necessary.

a) What is the value of memory location 11Ah?
   ________________
   0003 h

b) What is the value of memory location 11Ch?
   ________________
   0008 h or 0005 h

   ________________
   0003 h

c) What is the value of memory location 11Eh?
   ________________
   0003 h

d) What is the value of memory location 120h?
   ________________
   000A h

e) What is the value of memory location 122h?
   ________________
   0002 h

f) What is the value of memory location 124h?
   ________________
   0004 h
### Initial Contents of Memory in hex:

<table>
<thead>
<tr>
<th>Location</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>011C</td>
</tr>
<tr>
<td>102</td>
<td>3122</td>
</tr>
<tr>
<td>104</td>
<td>1120</td>
</tr>
<tr>
<td>106</td>
<td>011A</td>
</tr>
<tr>
<td>108</td>
<td>211E</td>
</tr>
<tr>
<td>10A</td>
<td>111A</td>
</tr>
<tr>
<td>10C</td>
<td>4112</td>
</tr>
<tr>
<td>10E</td>
<td>1120</td>
</tr>
<tr>
<td>110</td>
<td>1124</td>
</tr>
<tr>
<td>112</td>
<td>011C</td>
</tr>
<tr>
<td>114</td>
<td>211A</td>
</tr>
<tr>
<td>116</td>
<td>111C</td>
</tr>
<tr>
<td>118</td>
<td>7000</td>
</tr>
<tr>
<td>11A</td>
<td>0000</td>
</tr>
<tr>
<td>11C</td>
<td>0005</td>
</tr>
<tr>
<td>11E</td>
<td>0003</td>
</tr>
<tr>
<td>120</td>
<td>0000</td>
</tr>
<tr>
<td>122</td>
<td>0002</td>
</tr>
<tr>
<td>124</td>
<td>0004</td>
</tr>
</tbody>
</table>

### Rough Work

#### Step No:

1. AC = 0005
2. AC = 000A
3. location 120: 000A
4. AC = 0000
5. AC = 0003
6. location 11A: 0003
7. PC = 112
8. AC = 0005
9. AC = 0008
10. location 11C = 0008

**OR Alternatively**

7. PC = 11C

8. Can not continue, as we don’t know the contents of location 005
Q22. [10 marks] An 8-bit microprocessor has a total addressable space of 64 kBytes. Organize the memory space so that the first 16 kBytes is ROM. The next 8 kBytes is empty for future expansion. This is followed by 16 kBytes RAM. The remaining space is once again empty for future expansion.

(a) Label the starting and ending addresses of each region in hexadecimal. [4]

<table>
<thead>
<tr>
<th>Region</th>
<th>Start Address</th>
<th>End Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>16 k Bytes (ROM)</td>
<td>0000 h</td>
<td>3FFF F h</td>
</tr>
<tr>
<td>8 k Bytes (free)</td>
<td>4000 h</td>
<td>5FFF F h</td>
</tr>
<tr>
<td>16 k Bytes (RAM)</td>
<td>6000 h</td>
<td>9FFF F h</td>
</tr>
<tr>
<td>free</td>
<td>A000 h</td>
<td>FFFF F h</td>
</tr>
</tbody>
</table>

Rough Work

If the available ROM chips are (16k x 8) and RAM chips are (16k x 8). Draw a block diagram of a suitable memory design implementing the memory map described in part (a) on the reverse side of this sheet. Label the address, data, and control lines on the microprocessor and memory chips. You may use additional logic chips if required. [6]
Q 23. [15 marks] A microprocessor has an on-chip 2-way set associative cache with a total capacity of 8 kByte. Each line in cache can store sixteen 8-bit words. It has a total addressable space of 16 MBytes.

Consider the following CPU instruction:

**Load AB1234h**

a) Which set number will this request be stored in cache? [2]

<table>
<thead>
<tr>
<th>Number of lines in a set ((k)) = 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total capacity of the cache (= 8 \text{ k Byte} )</td>
</tr>
<tr>
<td>Block size (= 2^w=) line size = sixteen 8 bit words = 16 bytes</td>
</tr>
<tr>
<td>Number of sets ((v) = 2^d = \text{Total capacity } / (k \times \text{line size}) = 256 )</td>
</tr>
<tr>
<td>Total addressable space (= 2^{s+w} = 16 \text{ M bytes} )</td>
</tr>
<tr>
<td>Address length (= (s+w) \text{ bits} = 24 \text{ bits} )</td>
</tr>
</tbody>
</table>

Thus, in the main memory address:

- Size of set \(= d = 8 \text{ bits} \)
- Size of a word \(= w = 4 \text{ bits} \)
- Size of a tag \(= (s-d) \text{ bits} = 12 \text{ bits} \)

Therefore, **Load AB1234h** will store the result in set number: **23 h**

b) What will be its corresponding tag number? [2]

The corresponding tag number will be: AB1 h

If the microprocessor had a direct mapped cache with half the capacity of the set associative cache. Then,

c) Which line number would the request be saved in? [2]

If a microprocessor had a direct mapped cache with half the capacity of the set of the two-way set associative cache, then it would have a capacity of 4 kbytes and effectively the same number of lines as the number of sets in the set associative cache, i.e.

<table>
<thead>
<tr>
<th>Number of lines in cache (= m = 2 = 256 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thus, in the main memory address:</td>
</tr>
<tr>
<td>Size of a line (= r = 8 \text{ bits} )</td>
</tr>
<tr>
<td>Size of a word (= w = 4 \text{ bits} )</td>
</tr>
<tr>
<td>Size of a tag (= (s-r) \text{ bits} = 12 \text{ bits} )</td>
</tr>
</tbody>
</table>

Therefore, **Load AB1234h** will store the result in line number: **23 h**
d) What will be the corresponding tag number? [2]

The corresponding tag number will once again be: \( \text{AB1h} \)

Now consider the case if the microprocessor had an associative cache with four times the capacity of the direct mapped cache.

e) Which line number would the request be saved in? [1]

In associative mapping any block can be stored in any line depending upon the current occupancy and replacement algorithms. Thus, we cannot say for sure which line number the request will be saved in.

f) What will be the corresponding tag number? [2]

Address length = \((s + w)\) bits = 24 bits

Thus, in the main memory address:

- Size of a word = \( w = 4 \) bits
- Size of a tag = \( s \) bits = 20 bits

Therefore, Load A B1234h will store the result with a tag number: \( \text{AB123h} \)

g) Which of the above caches will have the fastest access time. Justify your answer. [2]

The direct cache will have the fastest access time, given a hit, i.e. the presence of the requested block in cache as only one comparison of the tag will determine whether the requisite block is present or absent. Whereas, in associative memory one would have to compare the required tag with all tags in cache and in a set associative cache one would have to compare the required tag with all tags in the set.

h) Which of the above caches has the best overall performance considering replacement. [2]

Of the three schemes under consideration, the set-associative cache will have the best overall performance considering replacement. A access time from main memory is typical much larger than the access time from cache. Thus, if a situation where more than one block was alternatively required that mapped to the same line number. Direct mapping would require us to overwrite the blocks alternatively and this would be highly inefficient (thrashing). Associative mapping overcomes this but one has to go through the entire cache comparing tags to search for a block. Set-associative cache requires a relatively short time to access a block because one only checks for a required block by comparing tags within a set usually of size 2 or 4 whilst avoiding thrashing.
24. [15 marks]

a) Convert the binary number 1001011.101 to decimal [2]

\[
1001011.101 = 1 \times 2^{-3} + 1 \times 2^{-1} + 1 \times 2^0 + 1 \times 2^1 + 1 \times 2^3 + 1 \times 2^6 \\
= 0.125 + 0.5 + 1 + 2 + 8 + 64 \\
= 75.625
\]

b) Convert the hexadecimal number AC to decimal: [1]

\[
AC = 12 \times 16^0 + 10 \times 16^1 \\
= 172
\]

What are the smallest and largest numbers that you can represent in a 10-bit register using:

c) Unsigned notation [2]

- smallest number = 0
- largest number = \(2^{10} - 1 = 1024 - 1 = 1023\)

d) Two’s complement notation (using the left-most bit as the sign-bit) [2]

- smallest number = \(-2^9 = -512\)
- largest number = \(+2^9 - 1 = +511\)

e) Consider the following data definitions in a program: [2]

```
.data
list WORD 13, 11o, 42h, 00101011b
```

Fill out the corresponding values in HEX in the following table. The offsets are from the data segment.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>00 h</td>
</tr>
<tr>
<td>0001</td>
<td>0D h</td>
</tr>
</tbody>
</table>
Using the Little Endian format

0002  00 h
0003  09 h
0004  00 h
0005  42 h
0006  00 h
0007  2B h

f) Under what conditions are the following flags set:

i) **overflow flag**  [1]

- Two positive operands were added and their sum is negative
- Two negative operands were added and their sum is positive

ii) **carry flag**  [1]

If the result of an unsigned addition is too large (or too small) for the destination operand the carry flag is set.

---

Using the Little Endian format

**MOV AX, 0000h** ;clear the AX register

This MOV instruction does not affect the zero flag. Flags are affected by ALU operations.

---

h) Is it possible for a NEG instruction to set the Overflow flag?  [1]

The NEG instruction can set the overflow flag if we try to negate the smallest signed integer in a register. The NEG instruction results in a positive integer, which cannot fit into the same register. For example moving -128 to AL and trying to negate it results in +128, which cannot fit in AL and consequently causes the overflow, flag to be set.
i) Consider the following code:

```
mov ax, 0h
mov cx, 0Ah
doLoop:
    dec ax
    loop doLoop
```

What is the value of the ax register after the completion of the doLoop? [2]

The value of the ax register after completion of the loop is: **FFF6 h**
Part C [15 marks]

26. Which technique frees up the CPU by transferring data directly between the I/ O device and memory: [1]
   a. Programmed I/ O
   b. Interrupts
   c. I/ O Channel
   d. I/ O Registers
   e. I/ O Processor
   f. DMA

27. In which technique, does the CPU issue an I/ O command, continues to execute subsequent instructions, and is interrupted by the I/ O module when the latter has completed its work: [1]
   a. Programmed I/ O
   b. Interrupts
   c. I/ O Channel
   d. I/ O Registers
   e. I/ O Processor
   f. DMA

28. When an interrupt occurs, arrange the following operations in their order of occurrence? [3]
   a) interrupt service routine executed
   b) the registers are restored by popping their values off of the stack
   c) the processor identifies the source of the interrupt
   d) the program counter and other registers' values are pushed onto the stack
   e) the address of the interrupt service routine is placed in the program counter

   Order of Occurrence:

   1. _____ c _____
   2. _____ d _____
   3. _____ e _____
   4. _____ a _____
   5. _____ b _____
[10 marks]

Consider that an interrupt takes place when the microprocessor is executing an instruction stored at location N. The status flags at this instant contain the value ABCDh. The corresponding interrupt service routine (ISR) is stored in memory starting from location X and X + C (both inclusive)

a) What is the address of the instruction that will be executed after the interrupt occurs? [2]

\[ X \]

b) Where will the program branch to after the completion of the interrupt? [2]

\[ N + 1 \]

c) How will the microprocessor know where to branch after the completion of the interrupt? [3]

Prior to branching to the ISR the microprocessor stores the address of the current value of the Instruction pointer onto the stack. On completion of the subroutine it pops this address off the stack and loads this into the instruction pointer and thus branches back to the correct location.

d) What will happen if a second interrupt takes place while the first one is being serviced? [3]

If a second interrupt took place, then if it were of higher priority and the interrupt process was enabled the microprocessor would once again push the current context onto the stack and branch to the new ISR and on completing it branch back and complete the first ISR by popping off the context from the stack. If the interrupt process was disabled or if the new interrupt was of a lower or the same priority as the current interrupt it would remain pending.