# State Final Examination (Computer Science Sample Questions)

Summer 2022

## 1 Pushdown automata (3 points)

1. Write down the definition of a pushdown automaton.
2. Let us have the alphabet $\Sigma = \{+, *, (, )\}$. Consider the languages generated by the following context free grammars $G_A$, $G_B$

   $G_A = (\{A, E\}, \Sigma, \{A \to E + E, E \to EE \mid (E) \mid \lambda\}, A)$,

   $G_M = (\{M, E\}, \Sigma, \{M \to E * E, E \to EE \mid (E) \mid \lambda\}, M)$.

   Design a pushdown automaton that accepts the language $L(G_A) \cup L(G_M)$ by a final state, i.e. it accepts words generated by any of the grammars $G_A$, $G_M$.

## 2 Binary search trees (3 points)

1. Define a binary search tree.
2. Implement insertion of a new item to the tree (with no balancing). Analyze time complexity in the best and worst case.
3. Implement finding a successor: for a given key $x$, find a node with the smallest key greater than $x$. Analyze time complexity in the best and worst case.

Preferably use pseudocode for implementation.

## 3 Databases (3 points)

1. Explain the concept of *functional dependency of relation attributes* in relational data model. Is the relation $R(\underline{K}, A, B, C)$, with the key $(K)$, where $F = \{K \to AC,\ A \to B\}$ appropriately designed? If it is, explain why. If it is not, explain why and propose, how the design should be changed.
2. Consider the following relational database schema:

   *Company*(CompanyID, CompanyName, Country)
   *Product*(ProductID, CompanyID, ProductName, ProductType)
   *Product*.CompanyID $\subseteq$ *Company*.CompanyID

   Underlined are the relation keys.

   - Using an SQL query, find companies that don't produce any product of type 'Car'.
   - How would the semantic meaning of the model change (if at all) if the key of the relation *Product* changed from ProductID to concatenated key ProductID,CompanyID?

3. Is it guaranteed that schedules
   `S_1=(R_1(X),W_2(X),W_1(Y),W_2(Y),COMMIT_1,COMMIT_2)` and
   `S_2=(R_1(X),W_1(Y),COMMIT_1,W_2(X),W_2(Y),COMMIT_2)` will do the same change in the database?

   Explain your decision.

# 4 CPU emulator (3 points)

Your task is to design internal interfaces for a software emulator of a simple fictional processor. The processor has 16 32-bit registers, a 32-bit instruction pointer, and a memory representing a contiguous block of 8-bit bytes with an initial address of 0 and maximum size of 2GB. Both code and data are stored in this memory; your task does not include the initialization of memory and register content.

For each instruction the first byte (*opcode*) specifies the instruction itself, and is optionally followed by more bytes that hold the parameters of the instruction – the total byte count and the meaning of each byte is uniquely determined by the opcode. Some opcodes may remain undefined – an attempt to execute such an instruction terminates the emulator with an appropriate error message. Other possible errors should be treated similarly, such as an attempt to execute an instruction located (even if only partially) outside of the memory block or an attempt to read or write data to a position (partially) outside of the memory block.

The emulator must be easily extensible with new instructions (by attaching new source files implementing the new instructions together with minimal modification of the existing code). The interface of the common part of the code should make the implementation of individual instructions as easy as possible.

The proposed interface should also be usable for emulation of a multiprocessor system in which processors running in parallel with separate registers and shared memory are emulated (by a single-threaded emulator, i.e. without the need for synchronization).

1. Define the complete interface (in the form of classes and headers of their public functions) between the common part of the emulator and the implementation of an instruction. It includes
   - The mechanism for registering the instruction implementation under the given opcode in the emulator
   - The header of the function executing the given instruction and (if necessary) other functions from the instruction interface, called by the emulator
   - Headers of common emulator functions called from the code implementing individual instructions

   Describe how the error situations (described above) will be resolved.
2. Write a complete implementation of these instructions:
   - `STORE32 ra,rb,c32`, which writes the 32-bit contents of the `ra` register as 4 bytes into memory at the address given by the sum of the contents of the `rb` register and the `c32` constant (and into the following 3 bytes). The `ra` register number is stored in the lower 4 bits of the first byte after the opcode, the `rb` register number in its upper 4 bits and the `c32` constant is in the next 4 bytes, so the instruction has a total of 6 bytes.
   - `JEQ ra,rb,c32`, which compares the contents of registers `ra` and `rb` and, if they are identical, jumps to the address specified by the constant `c32`, otherwise the execution continues with the following instruction. The instruction parameters are coded in the same way as for `STORE32`. Explain how the jump instructions will be handled in the common part of the emulator.

   Also provide the code that will include these instructions in the emulator and describe where this code is called from.
3. The emulated processor should also run in a mode where 32-bit numbers are stored in the opposite format (*little/big-endian*) than the environment in which the emulator is running. How do you solve this problem?

To solve the task, choose any language from the set {Java, C#, C++}.

# 5 Language construct compilation (3 points)

Consider a simple processor with a MIPS-inspired architecture. The processor has 32 32-bit general-purpose integer registers, denoted **$R0**-**$R31**, and one 32-bit register **PC**, which points to the beginning of the next instruction. The value of the **$R0** register is always **0**.

The following table lists the instructions supported by the processor. We use the following operands in the description:
– *label* - symbolic code label
– *[mem]* - memory address
– *reg* - one general-purpose register

| instruction | description | operation |
|---|---|---|
| **LD** reg=[mem] | reads an integer variable from *mem* into the *reg* register | reg = *mem |
| **ST** [mem]=reg | stores the *reg* register in an integer variable at *mem* to | *mem = reg |
| **ADD** reg1=reg2,reg3 | add *reg2* register to *reg3* register and store result in *reg1* register | reg1 = reg2 + reg3 |
| **J** label | unconditional jump to the instruction address indicated by the label *label* | PC = label |
| **BEQ** reg1,reg2,label | conditional jump to be made if *reg1* and *reg2* registers are equal | if(reg1==reg2) PC = label |
| **BNE** reg1,reg2,label | conditional jump to be performed if *reg1* and *reg2* registers are not equal | if(reg1!=reg2) PC = label |
| **SLT** reg1=reg2,reg3 | sets the *reg1* register to 1 if the *reg2* register is smaller than the *reg3* register, otherwise it is set to 0 | reg1 = (reg2 < reg3) |

Consider the following sequence of instructions:

```
    LD    $r1=[q]
    LD    $r2=[n]
    LD    $r3=[inc]
    J     lab1
lab2:
    LD    $r4=[a]
    LD    $r5=[b]
    ADD   $r4=$r4,$r1
    SLT   $r5=$r5,$r4
    BEQ   $r5,$r0,lab3
    ST    [b] = $r4
lab3:
    ADD   $r1=$r1,$r3
lab1:
    SLT   $r4=$r1,$r2
    BNE   $r4,$r0,lab2
```

1. Chose (circle) all the C language snippets below that match the above instruction sequence (consider all unknown identifiers to be 32-bit integer variables that are declared somewhere). Write down the line numbers of the C language statements from the selected solution next to the corresponding instructions.

```
for (i=q; i<n; i+=inc)
  if (a+i <= b)
    b = a+i;
```

```
if (a+i > b)
  for (i=q; i<n; i+=inc)
    b = a+i;
```

```
for (i=q; i<n; i+=inc)
  if (a+i > b)
    b = a+i;
```

```
if (a+i < b)
  for (i=q; i<n; i+=inc)
    b = a+i;
```

```
i = q;
while (i<n) {
  if (a+i > b)
    b = a+i;
  i += inc;
}
```

None of the snippets matches. Write your own code and number the lines.

2. Write down (one line is sufficient) the change to the original C code that would result from replacing the `BEQ` instruction with a `BNE` instruction (with the same operands).
3. Suppose that the condition in the `if` statement is extended with `&& i < b`, i.e. the condition will read `if (`*original condition* `&& i < b)`. Update the original instruction sequence so that it matches the extended condition.

# 6 Networks (3 points)

Let us consider a network comprising 8 nodes denoted `A`–`H`. The dump of individual routing tables is in the following:

Node `A`

| Destination | Gateway | Interface |
|---|---|---|
| default | 10.0.1.3 | 10.0.1.5 |

Node `B`

| Destination | Gateway | Interface |
|---|---|---|
| default | 10.0.3.1 | 10.0.3.16 |

Node `C`

| Destination | Gateway | Interface |
|---|---|---|
| default | 10.0.4.1 | 10.0.4.2 |

Node `D`

| Destination | Gateway | Interface |
|---|---|---|
| 10.0.4.0/24 | on-link | 10.0.4.1 |
| default | 10.0.3.1 | 10.0.3.42 |

Node `E`

| Destination | Gateway | Interface |
|---|---|---|
| default | 10.0.1.4 | 10.0.1.9 |

Node `F`

| Destination | Gateway | Interface |
|---|---|---|
| 10.0.3.0/24 | on-link | 10.0.3.1 |
| 10.0.4.0/24 | 10.0.3.42 | 10.0.3.1 |
| default | 10.0.2.1 | 10.0.2.2 |

Node `G`

| Destination | Gateway | Interface |
|---|---|---|
| 10.0.1.0/24 | 10.0.1.2 | 10.0.1.1 |
| 10.0.0.0/16 | 10.0.2.2 | 10.0.2.1 |
| default | 192.168.0.1 | 192.168.0.42 |

Node `H`

| Destination | Gateway | Interface |
|---|---|---|
| 10.0.1.9/32 | on-link | 10.0.1.4 |
| 10.0.1.0/24 | on-link | 10.0.1.3 |
| default | 10.0.1.1 | 10.0.1.2 |

The route metrics are not included, assume they are all equal. Furthermore, assume that each network interface has exactly one IPv4 address. Loopback interfaces are omitted for the sake of simplicity.

1. Sketch a diagram outlining the network topology with IP addresses denoting individual network interfaces (the interfaces have no names, but the IP addresses identify them uniquely).

2. Consider the following IPv4 datagram sent from node `A`:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Ver = 4|IHL = 5|Type of Service|       Total Length = 1044     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Identification = 0xc001   |Flags|    Fragment Offset = 0  |     Flags:
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+     - do not fragment = 0
|TimeToLive = 3 |Proto = 17(UDP)|         Header Checksum       |     - more fragments = 0
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                  Source Address = 10.0.1.5                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               Destination Address = 10.0.4.2                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The values of fields that are not explicitly mentioned are irrelevant to our scenario, assume they are filled in correctly. Describe the traversal of the datagram through the network, in particular:

- list the nodes through which the datagram is routed,
- explain what each node does with the datagram (give reasons based on values from the datagram header and the routing tables),
- when the datagram is being routed, what properties of the datagram or the routing tables are being changed and how.

# 7 Set theory (specialization question – 3 points)

1. Define ordinal numbers.
2. Define cardinal numbers.
3. Is there a set of all ordinal numbers? Prove your answer.

# 8 Ramsey theory (specialization question – 3 points)

1. State Ramsey theorem for graphs.
2. Using Ramsey theorem prove the following:
   For every $k > 1$ there exists positive integer $M(k)$ such that for every partitioning of the set $\{1, 2, 3, \ldots, M(K)\}$ into two colours there exists monochromatic set of the form $\{x_1, x_2, x_3, \ldots, x_k, x_1 + x_2 + x_3 + \cdots + x_k\}$.

# 9 Edge colouring (specialization question – 3 points)

1. Define edge colouring of a graph.

2. State Vizing theorem.

3. Describe edge chromatic number of graph $K_n$ for every given positive integer $n$.