**Total Marks: 60** **Duration 3hrs** **Pages: 2**

**CS2011 Foundations of Computing Systems, End Semester Exam**

Nov 21, 2025

**Q 1:** The signal from a *water level sensor* can either be FULL, MID, or EMPTY. This question is about designing a *water pump controller* which interfaces with such a sensor and controls the water pump to ensure that the tank is filled whenever the tank gets empty.

**Input:** The signal from the sensor

**Output:** The signal to the pump which can be either ON and OFF

The controller *should satisfy* the following condition.

- The output signal should be turned OFF when the sensor gives out the signal FULL.
- The output signal should be turned ON when the sensor gives out the signal EMPTY.
- The pump once switched ON should not switch off till the tank is full.
- The pump should not be switched ON unless the tank is empty.

(a) (10 points) If we encode EMPTY, MID and FULL as 00, 01, and 11 respectively and the output OFF and ON as 0 and 1 respectively, describe control in the HDL.

**Q 2:** (15 points) Give a program in Hack assembly that computes the maximum element of an array of size N. Assume that the symbol `Arr` contains the starting address of the array and the symbol `N` contains the size of the array.

*Hint:* Have auxiliary variables MAX for the maximum element seen so far and Ptr for the "pointer" into the array element.

**Q 3:** (15 points) Write a function `sumN` for the Jack VM that takes as argument an integer n and computes the sum of the first n natural numbers. Your function should not just use the formula and should be iterative and not recursive.

**Q 4:** Consider the following hypothetical 16-bit RISC machine with 16 general purpose registers registers R0, ..., R15. There are no special registers and all the 16 registers can be used for arithmetic instructions, addresses during store and load as well as value registers for performing conditional jumps. The instruction set is summarized below.

**Arithmetic instructions** For operators being one of +, , or * and R, S, T being one of the 16 registers.

$$R := ST$$

**Load instruction** `LOAD R A` Load in register R the value at address stored in register A. Then R.

**Store instruction** `STORE R A` Store the value of the register R in the address stored in register A.

**Jump instruction** The conditional jumps all take a register R and transfer control based on the value of this register.

- `JMP LABEL` (unconditional jump)
- `JZ R LABEL`
- `JNZ R LABEL`
- `JGT R LABEL`
- `JGE R LABEL`
- `JLT R LABEL`
- `JLE R LABEL`

Besides these one can label program points using the syntax `LABEL:`.

(a) (3 points) Give an ocaml data type to represent the registers of the machine.

(b) (3 points) Give an ocaml data type to represent the operators supported by the machine (i.e. +, , and *).

(c) (3 points) Give an ocaml data type to capture the instructions of the machine.

(d) (3 points) Give an ocaml data type to capture assembly language programs.

(e) (3 points) Assuming all instructions are 16-bit, give a program to resolve the address of the labels in the program.

# A Virtual machine description

Here is the short description of the virtual machine instructions (relevant for the above problem).

- `push segment n`
- `pop segment n`
- Binary arithmetic operators `add`, `sub`. Takes top two values on stack and pushes the result of operation.
- Binary relational operators `lt`, `gt`, `ne`, and `eq`. Takes top two values and pushes the result of the comparison onto stack.

- Conditional jump `if-goto LABEL`. Takes the top value of the stack and jumps if true (i.e. 1).
- `goto LABEL` Unconditional jump to LABEL.

The segments can be any one of the three `argument`, `local`, and `constant` (other segments are irrelevant for this problem).