UNIVERSITY OF BRISTOL

PRACTICE PAPER

School of Computer Science

Examination for the Degree of
Master of Science in Computer Science (conversion)

COMSM1302
Overview of Computer Architecture

TIME ALLOWED:
2 Hours

This paper contains **eighteen** questions.
All questions will be marked.
The maximum for this paper is **100 marks**.

Other Instructions

1. **Make sure you carefully read any instructions on the question paper and answer booklet.**
2. **You must use the answer booklet to answer all questions.**
3. **This is a closed-book examination; books, notes, or other resources are forbidden.**
4. **A reference sheet will be given out alongside the question paper.**
5. **Any of the question paper can be used for rough work, but only the answer booklet will be marked.**
6. **A non-programmable calculator may be used for this examination.**

TURN OVER ONLY WHEN TOLD TO BEGIN WORK

Section 1 — Short-answer questions (50 marks total)

You do not need to justify your answers for any of the questions in this section, and you will not receive partial credit for showing your reasoning. Just write your answers down in the shortest form possible, e.g. "A" for multiple-choice questions, "True" for true/false questions, or "23" for numerical questions. If you do display working, circle or otherwise indicate your final answer, as if it cannot be identified then the question will not be marked.

Question 1 (5 marks)

(a) Write a single Hack assembly instruction that stores the value 32 in `A`.

(b) Write a single Hack assembly instruction that stores $-$`D` in `M`, then jumps if `D` $= 0$.

Question 2 (4 marks)

Which of the following memory segments in Hack VM would be most appropriate, when compiling from a high-level language such as C, to store a global variable?

A. `this`.

B. `that`.

C. `pointer`.

D. `local`.

E. `static`.

F. It depends on the context.

Question 3 (5 marks)

In an **assembler**, which of the following tasks would typically be carried out during lexing? You can pick more than one option.

A. Converting the program from a string into a sequence of tokens.

B. Constructing a symbol table mapping labels to memory addresses.

C. Constructing a symbol table mapping variables to memory addresses.

D. Converting each line of assembly into machine code.

E. Optimising the resulting machine code.

Question 4 (4 marks)

What are the binary values of the comp, dest, and jump operands for the C-instruction `AM=!M`? Your answers should include any leading 0s, where appropriate.

Question 5 (5 marks)

Which of the following are properties of an ISA, and which are properties of a microarchitecture?

A. Word length and address space of memory,

B. Energy efficiency.

C. Clock speed.

D. Number of transistors used.

E. Addressing modes supported.

Question 6 (4 marks)

What value will the following sequence of VM operations leave on top of the stack?

```
push constant 4
push constant 5
sub
not
push constant 0
eq
```

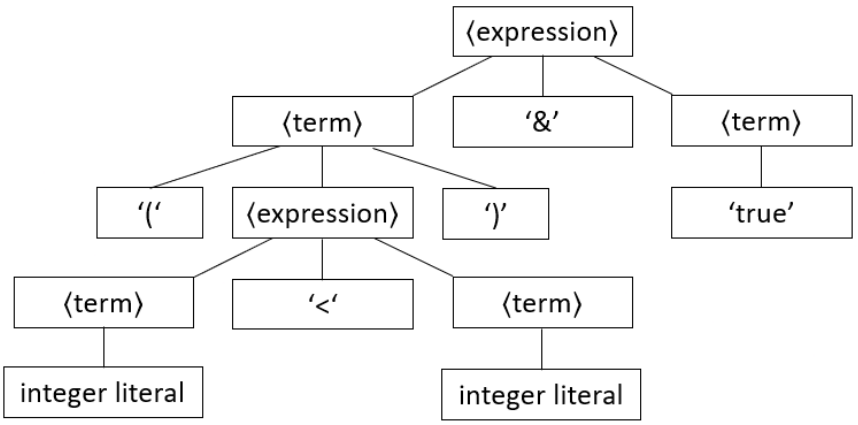
Question 7 (5 marks)

Below is part of the Jack grammar:

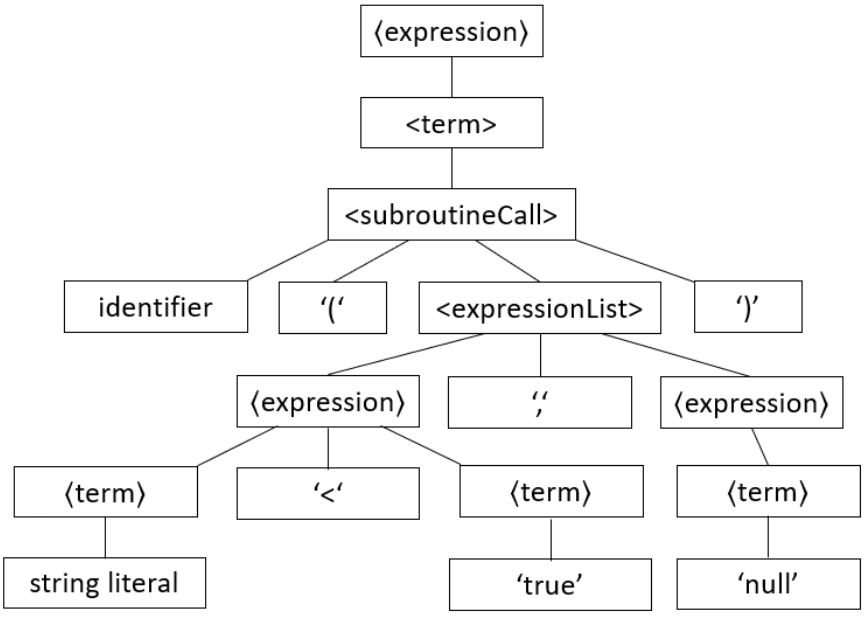
⟨expression⟩ ::= ⟨term⟩, {('+' | '-' | '*' | '/' | '&' | '|' | '<' | '>' | '='), ⟨term⟩};
⟨term⟩ ::= integerliteral | stringliteral | 'true' | 'false' | 'null' | 'this' | identifier, ['[', ⟨expression⟩, ']'] | '(', ⟨expression⟩, ')' | (('-' | '~'), ⟨term⟩) | ⟨subroutineCall⟩;
⟨subroutineCall⟩ ::= identifier, ['.', identifier], '(', ⟨expressionList⟩, ')';
⟨expressionList⟩ ::= [⟨expression⟩, {',', ⟨expression⟩}];

Which of the below is **not** a valid CST for a valid ⟨expression⟩?

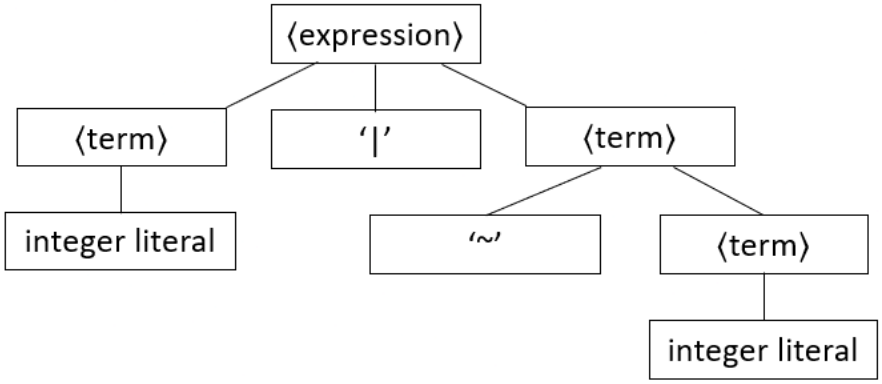
A.



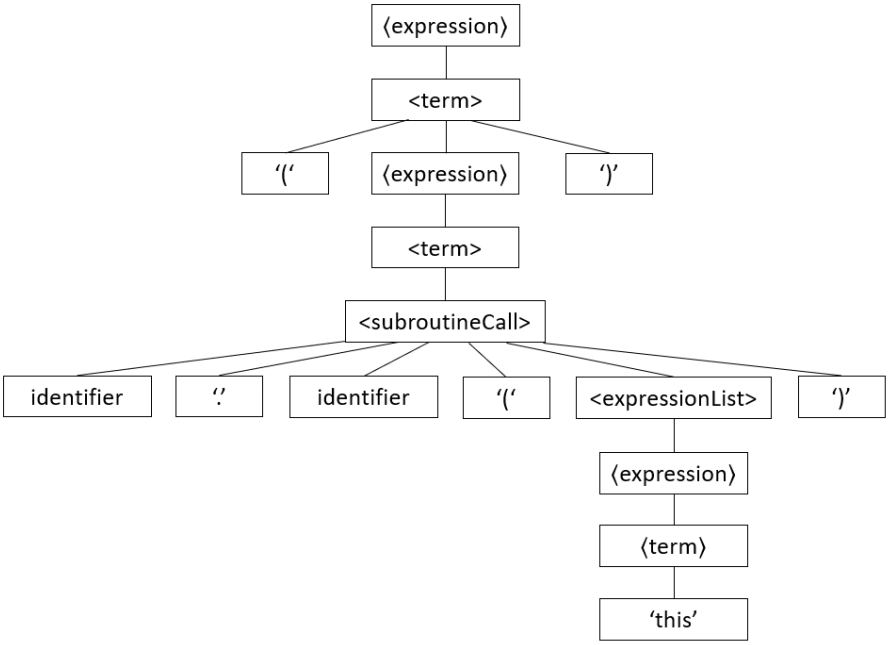
B.



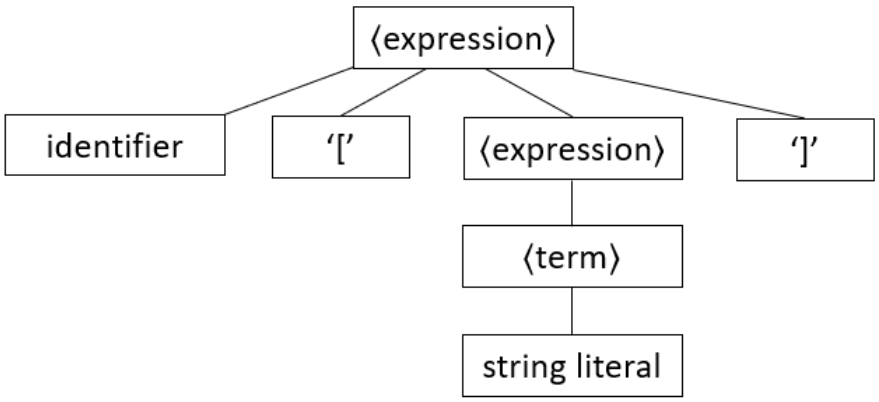
C.



D.

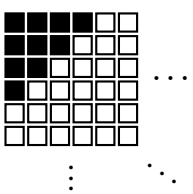


E.



Question 8 (5 marks)

In Hack, which values would need to be written to which RAM addresses in order to draw a right-angled triangle in the top-left corner of the screen as shown, with a height of four pixels? You should assume that initially, all memory is set to zero. You may give your answer in either binary, decimal, or hexadecimal form.



Question 9 (4 marks)

What is the technical term for each of the following concepts? For each part of the question, pick from one of the following options: "call frame", "global stack", "return address", "working stack", and "none of the above".

(a) The entire stack. **(1 mark)**

(b) A part of the stack dedicated to temporarily storing program state during a function call, to be restored on return. **(1 mark)**

(c) A part of the stack dedicated to storing values pushed directly via `push` Hack VM commands during the execution of a function. **(1 mark)**

(d) The value that will be left on top of the stack immediately after returning from a function. **(1 mark)**

Question 10 (4 marks)

Which **one** of the following statements is **false**?

A. EBNF can't be "simulated" using BNF — there are grammars written in EBNF that can't be rewritten in BNF on such a way that they accept the same expressions as valid.

B. In the grammar for a programming language, which strings are considered to be tokens depends on which language the grammar is for.

C. When writing a compiler, symbol tables are typically used to store information about functions and variables.

D. Given the grammar for a programming language and a parser for that grammar, it's always possible to tell which programs have valid syntax.

E. An abstract syntax tree may have fewer nodes than the corresponding concrete syntax tree.

Question 11 (5 marks)

Which RAM address (or addresses) does the Hack VM code below write the value 500 to?

```
push constant 200
pop this 0
push pointer 0
push pointer 1
add
pop pointer 1
push constant 500
pop that 7
```

You may assume the initial value of RAM[0] to RAM[4] are:

RAM[0] = 300;
RAM[1] = 290;
RAM[2] = 280;
RAM[3] = 1;
RAM[4] = 2;

Section 2 — Long-answer questions (50 marks total)

In this section, you should give the reasoning behind your answers unless otherwise specified. You **will** receive credit for partial answers or for incorrect answers with sensible reasoning.

Question 12 (7 marks)

(a) Explain what is meant by predication.

(b) Give an example of predication.

(c) Why can predication make code run faster?

Question 13 (15 marks)

Consider the Hack assembly code X which is a direct translation of the Hack VM instruction `return`. (So X assumes that the current contents of RAM are a state of the Hack VM under the standard memory map given on the reference sheet.)

(a) Explain in words how X "knows" which address to jump back to.

(b) Give **detailed pseudocode** for what X must do. This should include which values are written to which addresses of memory, and which order the writes happen in.

Question 14 (5 marks)

Explain in words how the Hack assembler is able to calculate the ROM address of each label during the lexing process.

Question 15 (5 marks)

Consider the following code in a fictional assembly language:

```
ADD #42, %35, $63
```

In this language, `ADD x, y, z` adds `x` to `y` and stores the value in `z`. The language uses `#` for immediate addressing, `$` for direct addressing, and `%` for indirect addressing. What will running this instruction do?

Question 16 (10 marks)

Write Hack assembly code to solve the following problem. Fill the first RAM[0] memory addresses counting upwards from RAM[256] with the values $0, 1, 2, \ldots$, so that if e.g. RAM[0] $= 3$ then the desired output is

RAM[256] $= 0$, RAM[257] $= 1$, RAM[258] $= 2$, RAM[259] unchanged.

It's OK if your code changed the value stored in RAM[0].

Question 17 (5 marks)

This question is about the first-fit memory allocation algorithm for Hack described in lectures that uses coalescence but not bins (i.e. "attempt 3"). We use -1 to denote free segments and null pointers, and 0 to denote used segments. Recall from lectures that in this algorithm, the structure of a segment's metadata is:

- The segment's first and last words contain its usable length.
- The segment's second word is -1 if the segment is free and 0 if it is used.
- If the segment is free, its third word is a pointer to the next free segment in the linked list and its fourth word is a pointer to the previous free segment in the linked list.

You are given the contents of the following memory addresses, which may or may not align with segment boundaries. Read them carefully.

RAM[0x800] = 0x2202;
RAM[0x801] = 0x9FC;
RAM[0x802] = 0xFFFF;
RAM[0x803] = 0xFFFF;
RAM[0x804] = 0x3003;
...
RAM[0x1200] = 0xDFD;
RAM[0x1201] = 0x0;
RAM[0x1202] = 0x2A02;
RAM[0x1203] = 0x2202;
...
RAM[0x21FE] = 0x844;
RAM[0x21FF] = 0x1FD;
RAM[0x2200] = 0x5FD;
RAM[0x2201] = 0xFFFF;
RAM[0x2202] = 0x3002;
...
RAM[0x29FF] = 0x1FD;
RAM[0x2A00] = 0x5FD;
RAM[0x2A01] = 0xFFFF;
...
RAM[0x2FFF] = 0x7FD;
RAM[0x3000] = 0xFFD;
RAM[0x3001] = 0xFFFF;
RAM[0x3002] = 0x803;

What segments are currently in memory? For each segment, you should give the first address, the last address, and whether it's free or used. If any segments can't be deduced from the information available, explain why.

Question 18 (3 marks)

Write **at most six** lines of Hack assembly that shift the contents of RAM[1] left by four bits.

THIS PAGE IS INTENTIONALLY LEFT BLANK

END OF PAPER