Test 2 practice paper A (theory)

This assessment contains questions that allow partial credit. Dismiss

12 OF 12 QUESTIONS REMAINING

Test Content

Question 1

2 Points

How many operands does a C-instruction have?

Add your answer

Integer, decimal or E notation allowed

Question 2

2 Points

What value does Hack VM use to represent "true" internally? Give your answer as a **signed** decimal value.

Add your answer

Integer, decimal or E notation allowed

Question 3

2 Points

Which of the following ISAs would you be most likely to see in a mobile phone?

- **A.** ARM
- **B.** Hack
- **C.** MIPS
- **D.** x64
- **E.** x86-64

Question 4

5 Points

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 5

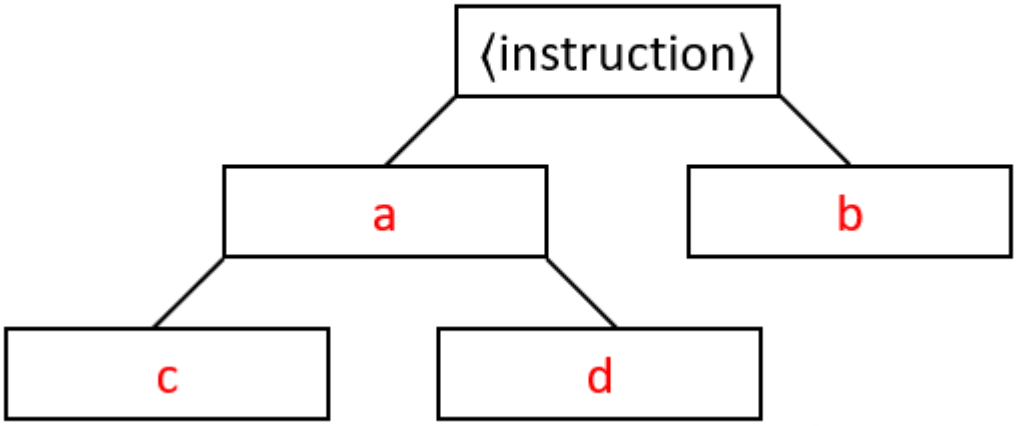
4 Points

The Hack assembly grammar is reproduced here for reference.

$$
\begin{aligned}
\langle \text{instruction} \rangle &::= (\langle \text{aInstruction} \rangle \mid \langle \text{cInstruction} \rangle),\ \text{newline};\\
\langle \text{aInstruction} \rangle &::= \text{“@”}, (\text{integerLiteral} \mid \text{identifier} \mid \langle \text{memoryKeyword} \rangle);\\
\langle \text{memoryKeyword} \rangle &::= \text{“SCREEN”} \mid \text{“KBD”} \mid \text{“SP”} \mid \text{“LCL”} \mid \text{“ARG”} \mid \text{“THIS”} \mid \text{“THAT”};\\
\langle \text{cInstruction} \rangle &::= [\langle \text{assignment} \rangle], \langle \text{computation} \rangle, [\langle \text{jump} \rangle];\\
\langle \text{assignment} \rangle &::= (\text{“A”} \mid \text{“D”} \mid \text{“M”} \mid (\text{“A”}, \text{“D”}) \mid (\text{“A”}, \text{“M”}) \mid (\text{“D”}, \text{“M”}) \mid\\
&\qquad\qquad (\text{“A”}, \text{“D”}, \text{“M”})), \text{“=”};\\
\langle \text{jump} \rangle &::= \text{“;”}, (\text{“JMP”} \mid \text{“JGT”} \mid \text{“JEQ”} \mid \text{“JLT”} \mid \text{“JGE”} \mid \text{“JNE”} \mid \text{“JLE”});\\
\langle \text{computation} \rangle &::= \text{“0”} \mid\\
&\qquad ([\text{“-”}], \text{“1”}) \mid\\
&\qquad ([\text{“-”} \mid \text{“!”}], (\text{“A”} \mid \text{“D”} \mid \text{“M”})) \mid\\
&\qquad ((\text{“A”} \mid \text{“D”} \mid \text{“M”}), (\text{“+”} \mid \text{“-”}), \text{“1”}) \mid\\
&\qquad ((\text{“A”} \mid \text{“M”}), \langle \text{binaryOp} \rangle, \text{“D”}) \mid\\
&\qquad (\text{“D”}, \langle \text{binaryOp} \rangle, (\text{“A”} \mid \text{“M”}));\\
\langle \text{binaryOp} \rangle &::= \text{“+”} \mid \text{“-”} \mid \text{“\&”} \mid \text{“|”};
\end{aligned}
$$

Using this grammar, complete the below parse tree.

@loopstart\n



a = , b = , c = , d =

Question 6

7 Points

Mark the following statements true or false:

a) A Hack VM program implemented on a Hack CPU under the standard memory mapping can never go 360 levels deep into recursive function calls without a stack overflow.

b) In order to compile labels, a typical Hack VM translator will keep careful track of the current ROM address in the produced assembly code.

c) Typically, manual heap allocation functions (such as C's malloc and free or Jack's Memory.alloc and Memory.deAlloc) do not prevent memory fragmentation.

d) A program calling malloc and free can manually defragment memory by freeing all malloc'd variables, then re-malloc'ing them.

e) Compilers for several different programming languages might use the same intermediate representation.

f) One intermediate representation might compile into several different assembly languages.

g) A function in Hack VM is expected to start with the non-".vm" part of its filename, followed by a ".", followed by a more descriptive name.

Question 7

3 Points

Fill in the blanks: The Hack instruction 0xE564 computes [ ], and stores the result in [ ] and [ ].

Question 8

6 Points

Memory allocated on the heap remains in use until [ ]. Variables can be allocated on the heap if their size is known [ ]. Memory allocated on the stack remains in use until [ ]. Variables can be allocated on the stack if their size is known [ ]. In C, we should normally expect variables to be allocated on [ ] unless an explicit call to malloc or free is made, in which case they will be allocated on [ ].

Question 9

4 Points

John is trying to put together a grammar that will match strings that include matching parentheses. For example, he wants "()" or "(())" or "((())())" to be valid under the grammar, but not "(" or "(()". He doesn't care about the empty string "". He puts together the following in BNF, with tokens of '(' and ')' :
<pair> ::= '(' ')'
<expression> ::= <pair> | '(' <expression> ')' | '(' <expression> ')' <expression>
Which of the following is correct?

- **A.** A string will be valid under the grammar if and only if it is a string of matching parentheses.
- **B.** Any string of matching parentheses will be valid under the grammar, but some strings of non-matching parentheses will also be valid. In other words, the grammar incorrectly declares some strings as "matching".
- **C.** Any string that's valid under the grammar is a string of matching parentheses, but not all strings of matching parentheses are valid under the grammar. In other words, the grammar incorrectly declares some strings as "not matching".
- **D.** None of the other options. In other words, the grammar incorrectly declares some strings "matching" and incorrectly declares other strings "not matching".

Question 10

5 Points

Suppose you have a list of binary words stored sequentially in RAM, starting at an address stored in the variable list_start. Which of the following code fragments will load the 50th item of the list into M?

A.
```
@list_start
A=M
D=A
@50
D=D+A
M=D
```

B.
```
@list_start
A=M
D=A
@50
D=D+A
A=D
```

C.
```
@list_start
D=A
@49
D=D+A
A=D
```

D.
```
@list_start
D=A
@50
D=D+A
A=D
```

E.
```
@list_start
A=M
D=A
@49
D=D+A
M=D
```

F. None of the other options.

Question 11

6 Points

Fill in the blanks in the following Hack VM program. Each Hack VM instruction contains 1-3 words. If you want to fill a line with an instruction containing only one word (like "add"), then choose "add", "N/A", and "N/A" from the drop-downs in that order. Likewise for instructions containing only two words.

// Add a character to the end of the given String object.

// The first argument will be the String's address. The second argument is the character to append (given as an integer).
// The String's fields are, in order:
// - The length of the String.
// - The maximum length of the String.
// - The base address of a segment of heap memory containing the characters that make up the String.
// Note that Jack strings are not null-terminated.
function String.appendChar 1
// Check whether string has space to append a new character.
// If not, crash by calling Sys.error(52).
push argument 0

push this 0
push this 1
lt

push constant 52

label nocrash
// Store address of new last character of string in local 0.
push this 2
push this 0
add

// Append character to string.
push local 0
pop pointer 1
push argument 1

// Update length
push this 0
push constant 1
add
pop this 0
// End of function
push constant 0

Question 12

4 Points

Consider the first-fit memory allocation algorithm discussed in lectures with coalescence of freed segments but without bins ("attempt 3"). Recall that in this algorithm, a value of "true" is used to denote a free segment and a value of "false" is used to denote a used segment. Here are the partial contents of three memory segments:

Segment 1:

RAM[0x759] = _____ (a)
RAM[0x75A] = 0xFFFF
RAM[0x75B] = _____ (b)
RAM[0x75C] = 0x2003
RAM[0x75D] = 0x1422
...
RAM[0x1105] = 0x9AB
RAM[0x1106] = _____

Segment 2:

RAM[0x1420] = _____
RAM[0x1421] = 0x0000
RAM[0x1422] = _____ (c)
RAM[0x1423] = 0x75C
RAM[0x1424] = _____
...
RAM[0x1500] = 0xDE
RAM[0x1501] = _____ (d)

Segment 3:

RAM[0x2000] = _____
RAM[0x2001] = 0xFFFF
RAM[0x2002] = _____ (e)
RAM[0x2003] = _____ (f)
RAM[0x2004] = _____
...
RAM[0x22FF] = _____ (g)
RAM[0x2300] = _____ (h)

Fill in the values of each of the following segments.

(a)
(b)
(c)
(d)
(e)
(f)
(g)

(h)

Questions Filter (12)

Save and Close

Submit