**How Computers Work-Final Review**

**I. Short Answer**

1. Write the number 2019 in binary

*A:*

2. How many bits would you need to represent a machine language with 15 instructions, 3 address modes? Assume that 20 bits are used for the address/parameter after the op code.

*A: you would need 4 bits for the instruction, 2 bits for the address mode, and the provided 20 bits for the value, for a total of 26 bits.*

3. What does the instruction register do?

*A: hold the current instruction copied from RAM*

4. Suppose you are inventing a new programming language. In order for the language to be understood by the computer, what other tool would you need to invent, specifically for this new language?

*A: A compiler/interpreter*

5. Suppose you have this line of machine code:

BRG 0110

If the line number of that instruction is 1000 (in decimal notation), what line, in decimal, will that instruction jump to if the accumulator is greater than 0? What line would it jump to if the accumulator is less than or equal to 0? Assume each instruction is 2 lengths apart.

*A: if greater, it jumps to 1006, if less than or equal to 0, it jumps to 1002.*

6. You walk in to Best Buy. The sign at the front says 1 TB of memory for $45. What type of memory is the sign likely referring to? You may assume for the purposes of this question modern capabilities of each type of memory.

*A: secondary memory. Secondary memory is usually on the order of TB, while RAM is on the order of GB.*

7. What are the three basic types of gates in a circuit?

*A: And, or, not*

8. What is an address mode?

A: A way to tell the computer how to handle the value that appears in the next section of bits

**II. Logo Design**

9. A classic: chutes and ladders. Suppose you are writing a program to play this game in a grid of 10 by 10 squares. You need to keep track of where each ladder and each chute starts and ends. You may assume that there are 6 chutes and 6 ladders anywhere on the board, starting and ending at arbitrary locations. You also need to keep track of where on the board both players are. Think about how you would represent the chutes and ladders board in Logo. What data structures would you use? What information would be stored in these data structures, and what would it represent? Draw your data structures and explain how your idea works.

A: Answers will vary. The basic idea here is to use two sets of 10 lists to represent the board, each list being 10 long. One set of 10 lists represents where on the board the chutes and ladders are. Label the top of a chute as A, C, E, G, I, K and the bottom of those chutes respectively as B, D, F, H, J, L. The ladders should be labelled at the bottom as M, O, Q, S, U, W and the tops of the ladders as N, P, R, T, V, X. This tells you where to move depending on which square you land on the board. Squares that are not labelled as either a chute or a ladder we will label with a dash The other set of 10 lists represents the two player’s positions on the board. So as to not confuse this set with the other set, we’ll use Y to represent player 1 and Z to represent player 2. The square that contains each player will be labelled appropriately. The rest of the squares will be dashes.

Take these two sets of lists for example:

[- A - - X - C - - -]

[K - - - - - I R - -]

[L - - B - - - - - -]

[- N G - - - Q - - J]

[- - - H V - E - - -]

[- - - - U - - - - -]

[- P D - - - - W - T]

[- - - - - - M - - -]

[- - O - - F - S - -]

[- - - - - - - - - -]

Other set of lists

[- - - - - - - - - -]

[- - - - - - - - - -]

[- - - - - - - - - -]

[- - - - - - - - - -]

[- - - - - - - - - -]

[- - - - - - - - - -]

[- - - - - - - - - -]

[- - - - - - - - - -]

[- - - - - - - - - -]

[- Z - - - - - Y - -]

Note that there are endless ways to solve this problem!

**III. Alan Turing**

1. What is Alan Turing’s main point in his article?

*A: Defining what it means for a machine to think*

2. Name and explain the nine objections to Turing’s argument

*A: (1) The Theological Objection*

*(2) The 'Heads in the Sand' Objection*

 *(3) The Mathematical Objection*

 *(4) The Argument from Consciousness*

 *(5) Arguments from Various Disabilities*

 *(6) Lady Lovelace's Objection*

 *(7) Argument from Continuity in the Nervous System*

 *(8) The Argument from Informality of Behaviour*

 *(9) The Argument from Extra-Sensory Perception (see article for descriptions)*

**IV. Assembly Language**

1. Write a program that takes three numbers and determines if there are any repeats. If any of them appear more than once, that number should be displayed and the program should stop. If there are no repeats, the three numbers should be printed back to the screen. Translate any three lines of your assembly code into machine language using the assembly language provided in class.

A: Here’s the basic idea without the assembly language, since there are numerous ways to write the assembly code:

Read in the first number. Simply store it in memory

Read in the second number. Compare it to the first number. If it is the same, output that number and stop the program. Otherwise, read in the third number. Compare it to the second number. If it is the same, output that number and stop the program. Otherwise, compare the third number to the first number, if they are the same output that number and stop the program, otherwise print to the screen all three numbers. Comparisons are relatively simple for this example. You would read one number in and subtract the other number. If the accumulator is equal to 0, the numbers are the same. If the accumulator is not equal to 0, the numbers are different. Here’s one way to do the problem in assembly code. The machine code is not written, since you could pick any three lines to translate from assembly to machine code.

1000 In 100 ; puts first number in 100

1002 In 200 ; puts second number in 200

1004 Load 100 ; puts num1 in AC

1006 Sub 200 ; AC, which has the value of num1, – num2

1008 BEQ (1026) ; AC = 0 which means num1 and num2 are equal, output and stop

1010 In 300; puts third number in 300

1012 Load 200 ; puts num2 in AC

1014 Sub 300 ; AC, which has the value of num2, – num3

1016 BEQ (1030) ; AC = 0 which means num2 = num3 so output and stop

1018 Load 100; AC is num1

1020 Sub 300; AC, which has the value of num1, – num3

1022 BEQ (1034); AC = 0 which means num1 = num3 so output and stop

1024 BR (1038) ; jump over the outputs and stopping because numbers are different

1026 Out 100 ; output num1

1028 END ; end program

1030 Out 200 ;output num2

1032 END ; end program

1034 Out 100 ; output num1

1036 END ; end program

1038 Out 100;numbers are different so output all three numbers

1040 Out 200

1042 Out 300

1044 END ; end program

**V. Logo Programming**

1. Write a Logo program called reverse, which takes in a string and the length of that string and outputs the string in the reverse order.

reverse “hello 5 should give Result: [O L L E H]

A:

TO reverse :alist :num

if (= count :alist 0) then op [] else op (se reverse bf :alist (- 1 :num) first :alist)

END

2. Write a logo program called grocery, which takes in a list of items, and returns the total price of all of those items. Assume that grapes are $1.20, oranges are $1.40, apples are $1.60 and pears are $2.00. Assume that the input comes in this form:

Grocery [apple apple orange grape orange grape pear]. This should return $10.40

A:

TO grocery :alist

if empty? :alist then op 0

if (= first :alist "apple) then op (+ 1.60 grocery bf :alist)

if (= first :alist "orange) then op (+ 1.40 grocery bf :alist)

if (= first :alist "pear) then op (+ 2.00 grocery bf :alist)

if (= first :alist "grape) then op (+ 1.20 grocery bf :alist)

END

**VI. Circuits**

1. Thanks to Geeksforgeeks.com for this problem!

A magnitude digital Comparator is a combinational circuit that **compares two binary numbers** in order to find out whether one binary number is equal, less than or greater than the other binary number. We logically design a circuit for which we will have two inputs one for A and other for B and have three output terminals, one for A > B condition, one for A = B condition and one for A < B condition.



A comparator used to compare two binary numbers each of two bits is called a 2-bit Magnitude comparator. It consists of four inputs and three outputs to generate less than, equal to and greater than between two binary numbers.

Draw a truth table for every possible outcome of values for 2 bits, and draw a circuit diagram to represent the truth table you made.

A: Will be available after review session (picture will be attached)