Exercise 2

Note, in an exam you will be expected to show all working i.e. intermediate stages or results for each question.

Question 1

(i) One hot encoding is a relatively simple method of encoding a number, each symbol is assigned a bit i.e. for a decimal number you require ten bits 0 – 9. To represent the number 35 you will require 20 bits:

<table>
<thead>
<tr>
<th>9 8 7 6 5 4 3 2 1 0</th>
<th>9 8 7 6 5 4 3 2 1 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0 0 0 1 0 0 0</td>
<td>0 0 0 1 0 0 0 0</td>
</tr>
</tbody>
</table>

Note, each digit is represented by 10 bits and only one bit is active at a time i.e. one hot encoded, this method simplifies encoding / decoding as only one bit is used per digit, the disadvantage is you need a lot of bits i.e. 20 bits compared to 6 bits if you used a binary representation.

(ii) An instruction must define the opcode (function), operands (data) and where any results will be stored.

(iii) LSB = least significant bit, MSB = most significant bit, Nibble = four bits, Byte = eight bits, Word = the standard memory unit used by a processor i.e. for an 8-bit processor it is a byte, for a 16-bit processor it is 16 bits, as most modern processors use 32 bits, this has become the normal interpretation.

Question 2

(i) When a logic 1 is placed on either of the inputs A or B the corresponding relay is energized causing its switch to close. When both the A and B switches are closed (the AND function) the third relay is the energised (the inverter function) causing its contact to switch, connecting the output to zero voltage i.e. a logic 0.

(ii) A circuit’s critical path is the longest series of logic gates. This worst case path through this combinational logic determines its performance i.e. how long it takes for an output to reach a new value in response to a change on its inputs. This is an important consideration when designing combinational logic circuits, as the longer these paths the slower the system clock must be i.e. the combinational logic circuit must be stable before it can be clocked into a register / flip-flop. The critical path for this relay circuit is therefore two gates i.e. relays associated with inputs A and B are in parallel giving a delay of one, their output driving the final stage.
(iii) Same as lab 1

An AND gate followed by a NOT gate is a NAND gate. The two input NOT gates can be implemented from two NAND gates with both inputs tied together.

Question 3

(i) To allow a computer to represent alphanumeric characters each symbol is assigned a binary number e.g. American Standard Code for Information Interchange (ASCII). Therefore, when a button is pressed on a keyboard its associated ASCII value is transmitted to the computer which then decodes this displaying the correct graphical character on the display e.g. 0x61 = ‘a’, i.e. each letter, number or character has been assigned a number, the absolute value of the number is not important, it is just used to identify each character. Therefore, when ASCII values are stored in memory the processor has no way of specifically identifying them as such i.e. the binary pattern “1100001” could be the character ‘a’ or the integer value 0x61, its up to the software to correctly decode / use the data it has been pointed to.

(ii) Machine code, binary or hexadecimal representations e.g.10101001 0001000. In general specific to a particular processor. Definitions

“Machine language, a pattern of bits encoding machine operations”
“The sequence of binary patterns that is executed by the hardware; the set of instructions that a computer’s CPU can understand and obey directly without any translation”

Assembler code, textual representations e.g. LDA 0x10. Definitions:

“A programming language that utilises symbols to represent operation codes and storage locations”
“Human readable notation for the machine language that a specific computer architecture uses, replacing raw binary patterns with symbols called mnemonics”

High level code, textual description e.g. IF (A=B) THEN C = C + 1. Definitions:

“A programming language where each instruction corresponds to several machine code instructions”
“A high level programming language is more user friendly, to some extent platform independent, providing a layer of abstraction between the programmer and the low level hardware”
In general assembly language programs allow you greater control over the processors internal / external hardware i.e. accessing the processors status register, external control registers in peripheral devices etc. It can also allow you to take advantage of possible parallelism allowing you to express instructions that can be performed in parallel. High level languages allow you to abstract away from the hardware, making coding easier to write, reducing program size and allowing code to be ported between processors i.e. a processor specific compiler is used to convert the high level code into processor specific assembler code.