Question 6. Software. **[4 marks]**

(a) [2 marks] Briefly explain the main reason why not all microcode has machine language equivalents and give an example.

Not all microcode makes sense in terms of useful functions. For example, you could have a microcode instruction that opened all the switches on the control lines preventing data from moving around the CPU.

(b) [2 marks] Briefly explain the main differences between a compiler and an interpreter.

Compiler takes source code and generates executable machine code that is then run. Interpreter reads source code and executes on line at a time. Compiled code doesn't require the compiler to run but interpreted code requires the interpreter to run.

Question 7. Understanding assembly language. [4 marks]

Consider the assembly language program below. Compute the final value of R0. What is the final value as a function of the initial value of R0 and R2.

Assume that R0 initially contains the value 5, R1 initially contains the value 1 and that R2 contains the value 4.

0: ADD R0 R0 R0
1: SUB R2 R2 R1
2: BZERO 4
3: BRANCH 0
4: HALT

R0*2^R2

R0 = 5 + 5 = 10
R2 = 4 - 1 = 3
R0 = 10 + 10 = 20
R2 = 3 – 1 = 2
R0 = 20 + 20 = 40
R2 = 2 – 1 = 1
R0 = 40 + 40 = 80
R2 = 1 – 1 = 0
R0 = 80

Key issues are that there is a loop here. One mark only was given for working out the formula.

Question 8. Writing assembly language. **[7 marks]**

(a) [5 marks] Write an assembly language program that computes the smaller of two values. The first value is stored in memory location 20, the second value is stored in memory location 21 and the result of the comparison should be stored in memory location 22.

```
0: LOAD R0 20
1: LOAD R1 21
2: SUB R2 R0 R1
3: BNEG 6
4: STORE 22 R1
5: HALT
6: STORE 22 R0
7: HALT

R0 = memory location 20
R1 = memory location 21
R2 = R0 – R1
If negative (R0 smaller than R1), go to instruction 6 and store R0 (the smaller) at 22
If positive or zero flag set (R1 smaller or equal to R0), store R1 at 22
```

(b) [2 marks] Construct the symbol table for the following assembly language program. Assume that code starts at memory location **10** and data at memory location **100**.

```
0: LOAD R0 a
loop:
1: ADD R0 R0 R0
2: STORE a R0
3: BRANCH loop
```

Symbol table:
a = 100 (note that data is stored from position 100 onwards)
loop = 11 (note that the code is stored from position 10 onwards so the second instruction is stored at 11)

Question 9. IO Management. **[5 marks]**

Consider the HACK computer architecture. Remember that this is a simple computer that uses memory mapped IO to communicate with the keyboard and the screen. Briefly describe the steps involved in implementing the following operation in a high-level language such as Java:

System.out.println("All mimsy were the borogoves,");

1. High-level program code calls println(s) method.
2. System class method println(s) calls operating system println(s) operation.
3. Operating system println(s) operation calls *hardware-specific* device driver printChar(c,r,ch) to implement the required functionality.
4. Assuming memory-mapped display, our device driver writes characters into appropriate memory locations.
5. I/O controller reads from display memory and drives the physical display via electrical signals.

Key things I am looking for here are: identification of the different layers and the different responsibilities of each layer.

Question 10. Memory management. **[5 marks]**

A friend has just bought a HACK computer system that support virtual memory management. It has two pages of 32Kb making a total of 64Kb of RAM free for application programs and a hard drive with a swap file that can hold up to six pages of data.

What is the maximum size of an application program that can be run on the computer in pages (explain the reasons behind your calculation)? Can you explain the effect that running many small application programs at once might have on the overall speed of his computer?

Maximum size = 2 + 6 = 8. Assuming that the program occupies all of the available RAM and stores the rest of application that is not being currently used in the swap file.

Many small programs indicates that the pages in RAM will all be used and the swap file will have to be used. Accessing the swap file requires work by the operating system to pause and switch pages. This is slow and is made worse by the fact that reading and writing is slow compared to accessing RAM.