Computer Design (2004) Paper 5 Question 2 (SWM)

Sketch Answer

(*a*) *I will assume that (a,b,i,n,t,one,zero) correspond to addresses in memory which will hold the variables a, b, i, n in the above program, t is a temporary location, one holds the integer 1 and zero holds the integer 0. Moreover, I will assume that initially a=b=1 and that n holds the loop count value.*

```
        one,zero,i              // for loop initialiser: i=1-0
        // first do loop test
loop:   n,i,t                   // t=n-i
        t,one,t,exit            // t=t-1=n-i-1, finished if negative
        // increment i
        zero,i,t                // t=0-i
        t,one,t                 // t=t-1=-i-1
        zero,t,i                // i=-t=i+1
        // a=a+b
        zero,a,t                // t=-a
        t,b,a                   // a=t-b=-a-b
        zero,a,a                // a=-a (i.e. a+b)
        // b=a-b
        a,b,b
        // jump around loop
        zero,one,t,loop

exit:
```

(*b*) *A big problem with the SIC instruction set is that all operands are fetched from the main memory (two reads) and are written back to the memory (one write) without making good use of a register file. The instructions are also long (4 machine words for the 4 addresses). By having separate instruction and data caches, the instruction fetch and data memory accesses could be pipelined, i.e. a two stage pipeline: fetch, execute/memory access. Four cycles would be required to perform each instruction. Each cycle a 1/4 of an instruction would be fetched. The sequence in the execute stage would then be:*

1. *read first operand*
2. *read second operand*
3. *perform the subtract*
4. *perform the branch if the subtract produced a negative number*

(*c*) *The code density for SIC code is very poor. Each instruction is very long (4 words) and performs little work. Often the 4th word of instruction simply points to the next instruction in the sequence in order to annul any possible branch. This is very wasteful.*

This question requires a knowledge of instruction sets presented in lectures 1–7.