1999 Paper 3 Question 7 SIMON MOORE

Computer Design - Ans A. (Part Ib and Diploma)

a) A MIPS figure merely tells one how may instructions can be executed per second rather than how much work is performed per second. The amount of work performed per instruction is highly dependent upon the particular instruction set. Furthermore, on a modern computer the performance of the memory interface has significant impact on the computer's performance which is not necessarily reflected in the MIPS rating.

b) Pipelining allows various steps involved in executing an instruction to be performed in parallel. However, the rate of progress from one step to another is dependent upon the slowest step. Consequently an instruction can be executed more quickly by a simple none pipelined processor but a pipelined processor can be executing parts of several instructions at once so the aggregate performance is better. Also, Latches are required between pipeline stages (steps) to store intermediate results. These cause additional latency.

c) Data hazards arise when an instruction in a pipeline requires the result of a previous instruction prior to the result being written back to the register file.
The classic RISC pipeline has two data hazards:

1. it takes two cycles from the execute stage before the result is written back to the register file

2. it takes one cycle from the memory access stage before the result from a load is written back to the register file

There are two further data hazards not mentioned in lectures: load to store and store to instruction fetch (self modifying code).

d) Feed-forward paths can be used to reduce or remove the effects of data hazards by forwarding recent results prior to them being available from the register file. For example, a feed-forward path could be inserted around the execute stage of the classic RISC pipeline to forward data before it reaches the write back stage. Consequently the pipeline would not have to stall.
A feed-forward path from memory access stage to the execute stage would reduce but not eliminate the need for pipeline stalls. (I would then expect an example.)

Computer Design - Qu B. (Part Ib and Diploma)

a) A branch instruction is typically taken at the execute stage of the classic RISC pipeline. The next instruction in the sequence has already been fetched resides at the decode stage of the pipeline. If the branch is taken then the instruction at the decode stage is not the target of the branch. We have two choices: either to remove the instruction at the decode stage causing a bubble, or to allow the instruction to be executed. In the latter case, the programming model is that the effects of a branch instruction appeared to be delayed since the instruction after the branch is always executed. This is known as a branch delay slot.

b) Without conditional instructions (assuming a and b are in register r1 and r2 and the result is to end up in r0):

```
            cmp r1,r2       ; status from r1-r2
            bgt skip1       ; if r1>r2 goto skip1
            mov r1,r0       ; r0:=r1
            b   skip2       ; goto skip2
     skip1: mov r2 r0       ; r0:=r0
     skip2:
```

With conditional instructions:

```
            cmp r1,r2       ; status from r1-r2
            movgt r1,r0     ; if r1>r2 then r0:=r1
            movle r2,r0     ; if r1<=r2 then r0:=r2
```

c) Subroutine calls not only branch to a target address but they also save some state. At the very least the PC is saved. ARM processor's branch-with-link (bl) instruction saves the PC in R14. The Intel processor's call instruction saves the PC on the stack.