

Example Questions for the Processor Architecture Course

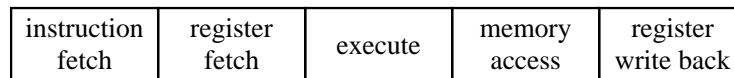
Simon Moore

initial release November, 1994

revised January, 1995

1. (a) What is the difference between a virtual and a physical address? [5 marks]
(b) Explain the principle behind virtual address translation. [10 marks]
(c) What does a translation look-aside buffer (TLB) do? [5 marks]

2. The classical RISC pipeline (e.g. used by the MIPS R2000 processor) is depicted below:



Using the above pipeline as a basis for discussion, explain the following terms and their performance implications:

- (a) load delay slots [10 marks]
 - (b) delayed branches [10 marks]
3. (a) Why is data locality exploited in processor designs? [5 marks]
(b) What is a register file and a cache, and how do they assist data locality? [15 marks]

4. The ARM *ADD* instruction has the following assembler syntax:

ADD{*cond*}{*S*} *Rd*, *Rn*, < *Op2* >

where < *cond* > is the conditional execution part
{*S*} sets condition codes if present
< *Op2* > = *Rm*, < *shift* > or < #*expression* >
Rd, *Rn*, *Rm* are registers
< #*expression* > is a constant expression
< *shift* > is < *shiftname* >< *register* >
or < *shiftname* > #*expression*
or *RRX* (rotate right one bit with extend)
< *shiftname* > is one of: *ASL* arithmetic shift left
LSL logical shift left (same as *ASL*)
LSR logical shift right
ASR arithmetic shift right
ROR rotate right

The ARM multiply and multiply-accumulate instructions have the following syntax:

MUL{*cond*}{*S*} *Rd*, *Rm*, *Rs*

MLA{*cond*}{*S*} *Rd*, *Rm*, *Rs*, *Rn*

where {*cond*} is the condition part
{*S*} sets the condition codes if present
MUL performs $Rd = Rm \times Rs$
MLA performs $Rd = Rm \times Rs + Rn$

And the ARM move instruction has the following syntax:

MOV{*cond*}{*S*} *Rd*, < *Op2* >

where {*cond*} is the condition part
< *Op2* > same as the definition for < *Op2* > for *ADD*

- (a) Why would the expression $a = b \times 3$ be compiled into an *ADD* instruction rather than a *MUL*? [3 marks]
- (b) What would be assembler for $a = b \times 3$ be if *b* was held in register *R1* and *a* was to be held in register *R2*? [3 marks]

Assuming multiplication was performed by a 2-bit Booth's algorithm, how would $a = b \times 83$ be compiled if optimisation was for:

- (c) Minimum code space? [7 marks]
- (d) Maximum speed? [7 marks]

N.B. for full marks both the code and an explanation are required.