Department of Computer Science & Engg.
COL 216 Computer Architecture
Minor Test – II

Date: 22.03.2017 Time: 09:30 – 10:30 Max marks: 15

1. ARM has multiply instructions MUL (multiply) and MULL (multiply long) that take two 32-bit integer operands and produce a 32 bit (in case of MUL) or a 64 bit (in case of MULL) result. Of course, MUL produces correct result only if it can be represented in 32 bits, wheras MULL gives correct result for all operand values. MULL has two variants - UMULL (for unsigned integers) and SMULL (for signed integers). However, MUL works for both signed as well as unsigned integers. Prove (or illustrate through adequate examples) that this indeed is true (that is, two separate instructions are not required). Consider operands that do not result in overflow.

(4)

Solution:

First we look at a few illustrative examples and then a general proof. For these examples, the word size is taken as 4 bits for simplicity. That means SMULL and UMULL produce 8 bit results and MUL produces 4 bit results. The table below shows a few examples showing what happens in SMULL and UMULL instructions. This is only to give a better insight and is not required as a part of the answer.

Operands	Interpretation as signed	Interpretation as unsigned	Result of SMULL	Result of UMULL
0011 x 0010	3 x 2 = 6	3 x 2 = 6	0000 0110	0000 0110
1101 x 0010	-3 x 2 = -6	13 x 2 = 26	1111 1010	0001 1010
1101 x 1110	-3 x -2 = 6	13 x 14 = 182	0000 0110	1011 0110
0011 x 0100	3 x 4 = 12	3 x 4 = 12	0000 1100	0000 1100
1101 x 0100	-3 x 4 = -12	13 x 4 = 52	1111 0100	0011 0100
1101 x 1100	-3 x -4 = 12	13 x 12 = 156	0000 1100	1001 1100

For MUL instruction, we can multiply the operands with any interpretation (signed or unsigned) and take the lower 4 bits of the result. Note that in the table above, results of SMULL and UMULL are identical in the lower 4 bits. In the table below we show the results of MUL and it can be observed that if there is no overflow, the results are correct for either interpretation.

Operands	Interpretation as signed	Interpretation as unsigned	Result of MUL	Result as signed	Result as unsigned
0011 x 0010	3 x 2 = 6	3 x 2 = 6	0110	6	6
1101 x 0010	-3 x 2 = -6	13 x 2 = 26	1010	-6	10 (overflow)
1101 x 1110	-3 x -2 = 6	13 x 14 = 182	0110	6	6 (overflow)
0011 x 0100	3 x 4 = 12	3 x 4 = 12	1100	-4 (overflow)	12
1101 x 0100	-3 x 4 = -12	13 x 4 = 52	0100	4 (overflow)	4 (overflow)
1101 x 1100	-3 x -4 = 12	13 x 12 = 156	1100	-4 (overflow)	12 (overflow)

Proof

Let the two operands be $X = (X_{31}, X_{30}, \ldots, X_1, X_0)$ and $Y = (Y_{31}, Y_{30}, \ldots, Y_1, Y_0)$.

If these are interpreted as unsigned integers, the product is $\Sigma_{i=0..31} (Y.X_i.2^i)$, whereas if these are interpreted as signed integers, the product is $-Y.X_{31}.2^{31} + \Sigma_{i=0..30} (Y.X_i.2^i)$.

The only differences between these two computations are as follows.

- The partial products $Y.X_i.2^i$ where i ranges from 0 to 30 are extended by zero for unsigned case and sign extended for the signed case to make up 64 bits.
- The term $Y.X_{31}.2^{31}$ is negated in case of signed interpretation.

It can be easily seen that zero extension or sign extension bits fall in the upper 32 bits of the partial products. Secondly, $Y.X_{31}.2^{31}$ and $-Y.X_{31}.2^{31}$ do not differ in the lower 32 bits (recall one of the rules to find 2's complement discussed in COL215: scanning from right to left, all initial zeroes and first 1 are same, the remaining bits are complemented). Therefore, the lower 32 bits of the product are same whether the operands are interpreted as signed or unsigned integers. (1)

If an integer falls within the range of 32 bits (signed or unsigned), its 32 bit representation can be obtained from its 64 bit representation simply by taking the lower 32 bits. (2)

Based on (1) and (2), we can conclude that the instruction MUL needs to simply multiply the two operands as either signed or unsigned integrs and output the lower 32 bits of the product. This will give correct result in either case.

2. Consider a datapath for implementing ARM instructions {DP, DT, MUL and MLA} using single cycle design approach shown as DATAPATH-1 on page 2. This figure is in an abstract form where the bit level details are omitted and multiplexers are considered as parts of the major modules shown. The delays of various modules are given in the table on right. RF has enough read/write ports to support all variants of the instructructions. Assume that all the modules provide 0-delay bypass paths. That is, if an instruction does not utilize a particular module, it goes through that module with zero delay. Find the maximum clock frequency for this datapath.

Module	Description	Delay
IM	Instruction Memory (read)	200 ps
RF	Register File (read or write)	140 ps
Shift	Shift/Rotate (combinational)	120 ps
Mult	Multiplier (combinational)	180 ps
ALU	Arithmetic-logical unit (combinational)	160 ps
DM	Data memory (read or write)	200 ps

(4)

Solution:

Since RF has enough ports, everything that needs to be read from RF is read concurrently in the beginning and everything that needs to be written into RF is written concurrently at the end.

Paths followed by different instruction and their variants are as follows.

DP (arithmetic/logical) without shift/rotate	IM => RF => ALU => RF
DP (arithmetic/logical) with shift/rotate	IM => RF => Shift => ALU => RF
DP (test) without shift/rotate	IM => RF => ALU
DP (test) with shift/rotate	IM => RF => Shift => ALU
DT (store) no shift/rotate, no auto inc/dec	IM => RF => ALU => DM
DT (store) no shift/rotate, auto inc/dec	IM => RF => ALU => DM => RF
DT (store) with shift/rotate, no auto inc/dec	IM => RF => Shift => ALU => DM
DT (store) with shift/rotate, auto inc/dec	IM => RF => Shift => ALU => DM => RF
DT (load) no shift/rotate, no auto inc/dec	IM => RF => ALU => DM => RF
DT (load) no shift/rotate, auto inc/dec	IM => RF => ALU => DM => RF
DT (load) with shift/rotate, no auto inc/dec	IM => RF => Shift => ALU => DM => RF
DT (load) with shift/rotate, auto inc/dec	IM => RF => Shift => ALU => DM => RF
MUL	IM => RF => Mult => RF
MLA	IM => RF => Mult => ALU => RF

Only the paths that are not dominated by other paths need to be considered. This leaves us with two paths -

IM => RF => Shift => ALU => DM => RF	delay = 200+140+120+160+200+140 = 960
IM => RF => Mult => ALU => RF	delay = 200+140+180+160+140 = 820

Minimum clock period for the datapath = longest delay = 960 ps
Maximum frequency = 1/.96 GHz = 1.04 GHz approx.

3. Registers reg1, reg2 etc. are introduced in DATAPATH-1 to get DATAPATH-2, as shown on page 2, for multi-cycle implementation of the instructions mentioned in Question 2. If an instruction bypasses a module, it bypasses the corresponding register as well. For example, if DM is bypassed, reg6 is also bypassed. Find the maximum clock frequency for this datapath. Find out whether this design will always be faster or always be slower than the design of Question 2, in terms of the total execution time of a program? Assume that registers reg1, reg2 etc have zero set-up time and zero output delay.

(4)

Solution:

Here the minimum clock period/maximum clock frequency is decided by the module with maximum delay.
Therefore, minimum clock period = 200 ps
Maximum clock frequency = 1/0.2 GHz = 5 GHz

Different instructions consume different number of clock cycles as shown below (one cycle per module).

DP (arithmetic/logical) without shift/rotate	4 cycles => 800 ps
DP (arithmetic/logical) with shift/rotate	5 cycles => 1000 ps
DP (test) without shift/rotate	3 cycles => 600 ps
DP (test) with shift/rotate	4 cycles => 800 ps
DT (store) no shift/rotate, no auto inc/dec	4 cycles => 800 ps
DT (store) no shift/rotate, auto inc/dec	5 cycles => 1000 ps
DT (store) with shift/rotate, no auto inc/dec	5 cycles => 1000 ps
DT (store) with shift/rotate, auto inc/dec	6 cycles => 1200 ps
DT (load) no shift/rotate, no auto inc/dec	5 cycles => 1000 ps
DT (load) no shift/rotate, auto inc/dec	5 cycles => 1000 ps
DT (load) with shift/rotate, no auto inc/dec	6 cycles => 1200 ps
DT (load) with shift/rotate, auto inc/dec	6 cycles => 1200 ps
MUL	4 cycles => 800 ps
MLA	5 cycles => 1000 ps

In Q2 all instructions take 960 ps. Here the instructions needing 3 or 4 cycles require 600 or 800 ps and are faster than instructions of Q2. The instructions needing 5 or 6 cycles require 1000 or 1200 ps and are slower than those of Q2. Therefore, the overall speed of this datapath as compared to the datapath of Q2 will depend on the fraction of different types of instructions in a program.

4. Consider a classical 5-stage pipeline to implement ARM instructions {DP, DT and B}. For DP instructions, no shift/rotate is allowed. For DT instructions, offset is always 0. When a branch instruction is detected, any instructions entered in pipeline after the branch instructions are flushed. When the branch completes the 3rd stage, its correct successor is fetched. Find the average number of clock cycles per instruction, if 20% instructions are branch instructions. Assume that there are no data hazards or structural hazards. How will this result change if the instructions after the branch are allowed to continue in the pipeline and are flushed only when the branch instruction reaches the 3rd stage and the condition for branching is found to be true?

(4)

Solution:

When instructions are flushed on detecting a branch instruction:

If a branch instruction is fetched in cycle t, the next instruction is fetched in cycle t+3. Otherwise, instructions enter the pipeline in every cycle.

Therefore, cycles per branch instruction = 3
Cycles per non-branch instruction = 1.
Since 20% instructions are branch instructions,
average number of cycles per instruction = 0.8 x 1 + 0.2 x 3 = 1.4

If instructions after branch instruction are allowed to continue,
the delay of 3 cycles is encountered only for the branches that are taken.

Assuming that fraction of the branches that are taken = p
average number of cycles per instruction = 0.8 x 1 + 0.2 x (3p + 1(1-p))
= 0.8 + 0.2 (2p + 1) = 1 + .4 p