## 

## **NATIONAL UNIVERSITY OF SINGAPORE**

## **Department of Electrical and Computer Engineering**

## **CG2028 Computer Organization**

**Tutorial 4 Solutions**

**Single Cycle Processor Design**

Questions in this tutorial assume the instruction format (i.e., architecture) and microarchitecture described in Lecture 4.

1. Write the assembly language instructions (consider the extended formats given in slides 37-40) corresponding to the machine codes below.
   1. 0x0224201C
   2. 0x0024201C
   3. 0x0404001C
   4. 0x0804001C

Ans:

* 1. 0x0224201C is EOR R2, R4, #0x1C
  2. 0x0024201C is MLA R2, R12, R0, R4. Note that I bit is 0 and M bit is 1 for MUL and MLA, whereas for usual DP Reg instructions, I bit is 0 and M bit is 0.
  3. 0x0404001C is STR R0, [R4], #-0x1C
  4. 0x0804001C is BEQ LABEL, where LABEL corresponds to the address of the instruction 6 instructions before the BEQ instruction.

1. Annotate the bit widths for all the connections of the microarchitecture given in Slide 28 of Lecture 4.

Ans:

CLK, PCS, PCSrc, FlagWrite (En), MemtoReg, MemWrite (WE), RegWrite (WE3), RegSrc, Z(Q), D are all 1-bit. If the ALU is designed to output only the Z flag, then ALUFlags is 1-bit. If all 4 flags are output by the ALU, of which Z flag alone is used, then ALUFlags is 4-bit.

op, ALUSrc are 2-bit.

cond, ALUControl, Rn (A1), Rm, Rd, A2, Rd (A3) are 4-bit.

funct is 6-bit.

imm8 is 8-bit.

All others connections are 32-bit.

1. Modify the microarchitecture given in Lecture 4 to incorporate BNE instruction. Detail the datapath and control unit modifications required, including logic expressions for new control signals / existing control signals which need to be modified, if any.

Ans:

No change required for the datapath. PCSrc control signal needs to be modified as given below.

PCSrc = PCS && CondEx;

where CondEx is derived using the block below.

switch(cond) {

case 0b0000 : //EQ

CondEx = (Z==1);

break;

case 0b0001 : //NE

CondEx = (Z==0);

break;

default :

CondEx = 1; //assuming the default is AL(always)

}

The logic above is given in C-syntax for ease of understanding. In real world, the equivalent statements in VHDL/Verilog is all you need to implement the Condition Check hardware.

The above block can be easily extended to include all the condition codes given in page 40. However, note that you will need an ALU capable of generating N, C and V flags, as well as 3 additional flip-flops (FFs) to store them. You will also need FlagWrite to be expanded to account for the fact that not all instructions affect all the flags (logical operations do not affect V flag, for example).

1. What is the range of instructions from the current instruction we can jump to using a branch instruction? Suggest how the architecture can be modified to increase this range. Briefly discuss the microarchitectural implications as well (detailed implementation is not expected).

Ans:

We have an 8-bit offset which is added to PC+4. However, the memory is byte-addressed, but each instruction takes up 4 bytes. Hence, the last 2 bits of the PC and offset are always zero for it to be meaningful, leaving us with effectively 6 bits to count instructions, which allows a range of 63 instructions to either side of PC+4 (note that the offset can be positive or negative depending on the U-bit). So from PC, we can jump by 64 instructions forward or 62 instructions backward.

One solution for increasing this range is to change the branch instruction format to make use of the unused bits. The range can be improved even further by encoding word address offset instead of a byte address offset, thereby not wasting the last two bits (this will increase the range 4 times). These improvements will need the Extend unit to be modified to extend the immediate differently for branch instructions (as compared to other instructions such as DP and memory type), as well as a control signal to tell the Extend unit which type of extension to perform.