國立交通大學 資訊工程系 計算機組織 第二次考試 (2014/5/29 10:10AM~12:20PM)

請注意：1. 不可看書及任何參考資料(Close book)。 2. 請詳述設計或計算過程(detail your design or computation process)。 3. 請依題號作答，否則高題號答案出現後，之後的低題號答案將不予計分(answer each problem in problem sequence. The answers of small ID problems that appear behind the answers of large ID problems will be ignored)。

1. **Arithmetic Division (8%)**

   Explain how non-restoring division and restoring division achieve the same result. You have to highlight the differences between two methods and derive the computation process to show two ways have the same result.

2. **Floating point operation (14%)**

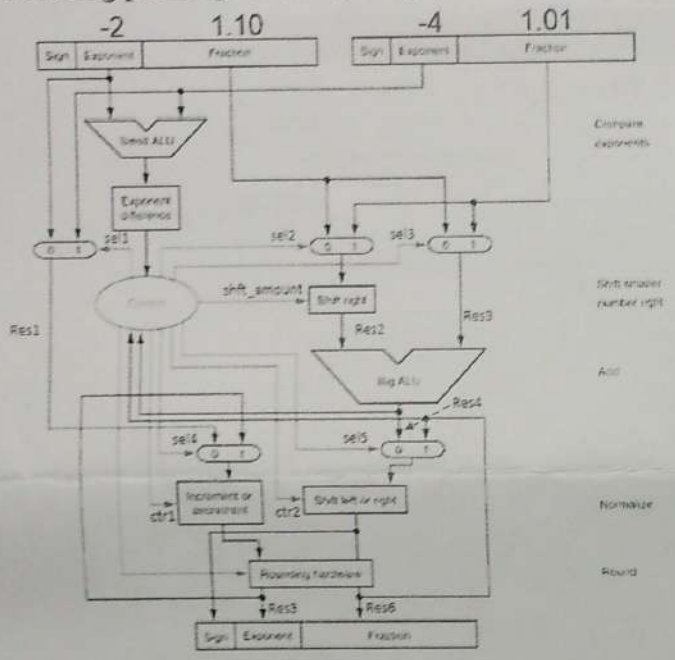


Figure 1

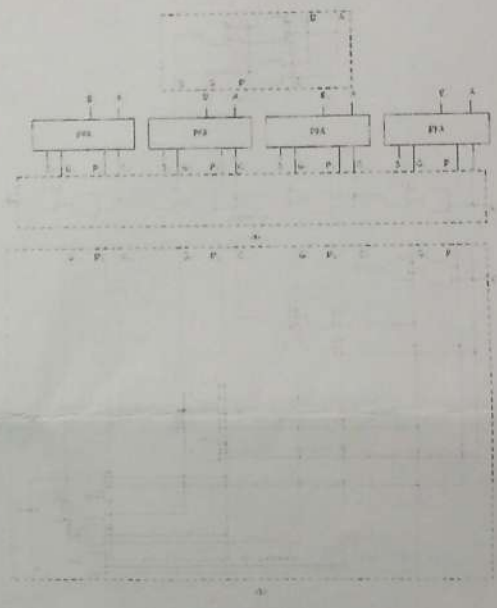


Figure 2

Figure 1 shows the hardware for floating adder/subtractor. Assume that we do not need to care about the order of input operands for the Big ALU, i.e., considering the operation of "*A*-*B*", it is both fine to make *A* go into the Big ALU from either the left side or the right side of input sources. Each fraction is rounded off to the second decimal place with the use of guard bit and round bit. Consider the operation of $1.10\times2^{-2} - 1.01\times2^{-4}$, please list the values of input selection (*sel*1 ~ *sel*5), computed results (*Res*1 ~ *Res*6), and other control signals (*shft_amount*, *ctr*1 and *ctr*2). Notably, *ctr*1 can be *Inc*, *Dec*, or *nop* and *ctr*2 can be *Lef*, *Rig*, or *nop*. For the process of normalization and rounding, you only need to list the values in the first run through this stage.

3. **Carry lookahead adder (13%)**

   a、 (4%) Figure 2 is a 4-bit carry lookahead adder. Assume that the delays of each AND-OR structure, XOR gate, and isolated AND gate are 3, 2, and 2 time units. What are the required delays of signals *carry out* and *sum* for the 4-bit carry lookahead adder?

   b、 (6%) Now we replace the 4-bit carry lookahead adder with a 2–level 2-bit carry lookahead adder. How many 2-bit lookahead adders do we need to use to achieve this goal (2%)? What are the

required delays of signals *carry out* and *sum* for the 2-level 2-bit carry lookahead adder (4%)?

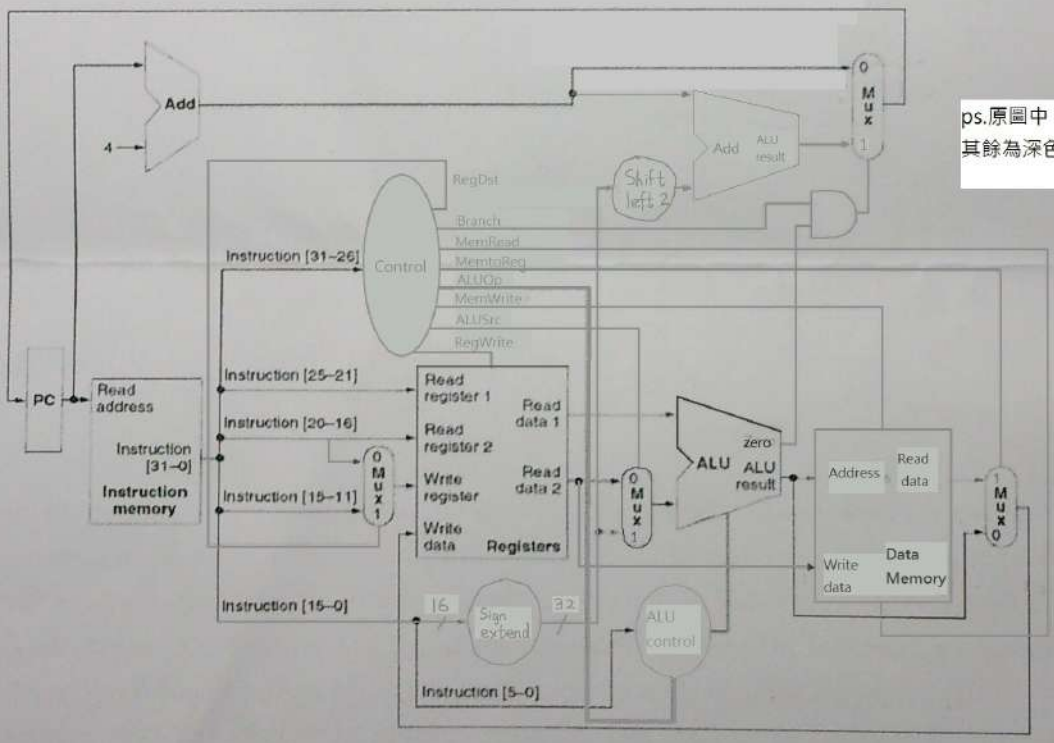
c、 (3%) Please explain why we prefer multi-level carry lookahead adder for practical use, e.g., a 64-bit adder (although we cannot see the benefit of multi-level carry lookahead adder from this example).

4. Single-Cycle CPU Design (22%)

Give the definition and formats of the instructions of a computer and the datapath for a single-cycle implementation of the computer as follows:

add	$rd, $rs, $rt	#$rd = $rs + $rt	R-format
lw	$rt, addr($rs)	#$rt = Memory[$rs + sign-extended addr]	I-format
sw	$rt, addr($rs)	#Memory[$rs + sign-extended addr] = $rt	I-format
beq	$rs, $rt, addr	#if ($rs = $rt) go to PC + 4 + 4 × addr	I-format

Name	Fields					
Field size	6 bits	5 bits	5 bits	5 bits	5 bits	6 bits
R-format	op	rs	rt	rd	shamt	funct
I-format	op	rs	rt	address/immediate		



ps.原圖中，control訊號是淺色線，其餘為深色線

Assume that control signals ALUOp = 00 for performing addition of the ALU unit, ALUOp = 01 for subtraction, and ALUOp = 10 while depending on the funct field of the instruction.

a、 (6%) Draw (highlight) the portion of the datapath used for fetching and executing instruction ***beq*** and specify the values of the control signals for this instruction in the above figure.

b、 (4%) Assume that logic blocks needed to implement the datapath have the following latencies: (Delays for other components are ignored.)

I-Mem	Add	Mux	ALU	Regs	D-Mem	Sign-extend	Shift-left-2
400ps	100ps	50ps	120ps	200ps	600ps	30ps	20ps

i. What is the critical path of the datapath and the clock cycle time of the implementation if we only had to support *beq* instruction?

ii. Repeat i. for *lw* instruction.

c、(9%) Please fill the control signals for each instruction in the following table:

Instruction	ALUSrc	ALUOp (2 bits)	MemRead	MemWrtie	MemtoReg	RegDst	RegWrite	Branch
R-format								
Lw								
Sw								

d、(3%) Suppose that a new instruction ***sltiu*** (set less than immediate unsigned) is added into the instruction set. Describe the modification of the datapath and control signals for the new instruction.

sltiu $rt, $rs, unsigned_immediate #if ($rs < unsigned_immediate), $rt = 1; else $rt = 0 (I-format)

5. Hazard, Forwarding and Code Re-Scheduling (30%)

```
// $t0 stores the starting address of array A[5] whose initial //
values are (0,1,2,4,6).
lw  $t1, 0($t0)
add $t3, $t1, $t2
lw  $t2, 4($t0)
bne $t1, $t2, tarAdr
sub $t4, $t1, $t3
sw  $t4, 8($t0)
tarAdr: add  $t5, $t1, $t4
sw  $t5, 16($t0)
add $t5, $t1, $t3
sw  $t5, 12($t0)
```

a、(6%) What are structure hazard, data hazard, and control hazard?

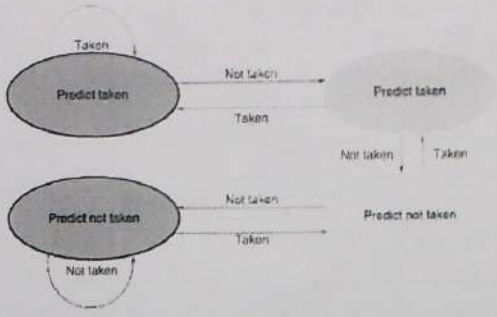
b、(8%) Consider the above code sequence and a five-stage pipelined CPU (IF, ID, EX, MEM, WB) with two forwarding schemes (forward the outcomes of EX and MEM to the inputs of ALU) and each branch instruction can know its target address and determines if it should branch to target address or not after ID stage. Before knowing the decision of branch instruction, sequential execution is assumed. Please use multi-cycle pipeline diagram to illustrate the execution process. You have to highlight the occurrence of flushing or forwarding in the diagram, and you also have to highlight the stall happening in a specific stage using two rows or repeatedly fetching instruction using two IF tagged marks.

c、(10%) Please reorder the code sequence to optimize its required clock cycles (4%). What are the

required clock cycles to complete the original and optimized code sequences (6%)?

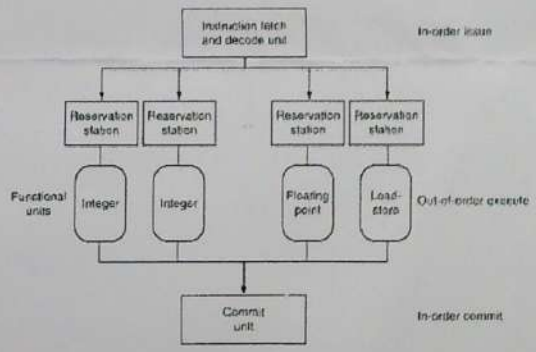
d、(6%) Hazard detection can be classified into two types, i.e., EX hazard and MEM hazard. The MEM hazard detection scheme is written as: if (MEM/WB.RegWrite and (MEM/WB.RegisterRd ≠0) and (MEM/WB.RegisterRd = ID/EX.RegisterRs)). Please refine the formula to a perfect version and explain why the original one is not perfect.

6. **2-bit Predictor (9%)**



Consider the above 2-bit predictor scheme and each loop has its own 2-bit predictor. If the nested loop has more levels of loops, does the prediction accuracy also increase (3%)? You have to explain your reasons but not only say "yes" or "no". If a code sequence has two levels of nested loops, the outer loop will iterate five times (i.e., 5 times "taken" and one time "not-taken") while the inner loop will iterate 6 times (i.e., 6 times "taken" and one time "not-taken"). What is the prediction accuracy as this loop is visited at the second time (6%)?

7. **SuperScalar CPU (10%)**



The above figure shows a superscalar CPU, please explain the operation of superscalar CPU.

8. **Loop Unrolling and Static Dual Issues (8%)**

```
Loop: lw   $t0, 0($s1)
      addu $t0, $t0, $t1
      sw   $t0, 0($s2)
      addi $s1, $s1,-4
      addi $s2, $s2,-4
      bne  $s1, $zero, Loop
```

Consider the above code sequence. Please use the technique of loop unrolling to duplicate this code sequences three times and optimally schedule the unrolled code sequences to a static dual issue pipelined MIPS CPU where one packet is for ALU/branch instructions and the other is for lw/sw instructions.