0907432 Computer Design (Fall 2015)

Final Exam

الاسم: الرقم التسلسلي: رقم الشعبة: 1

==

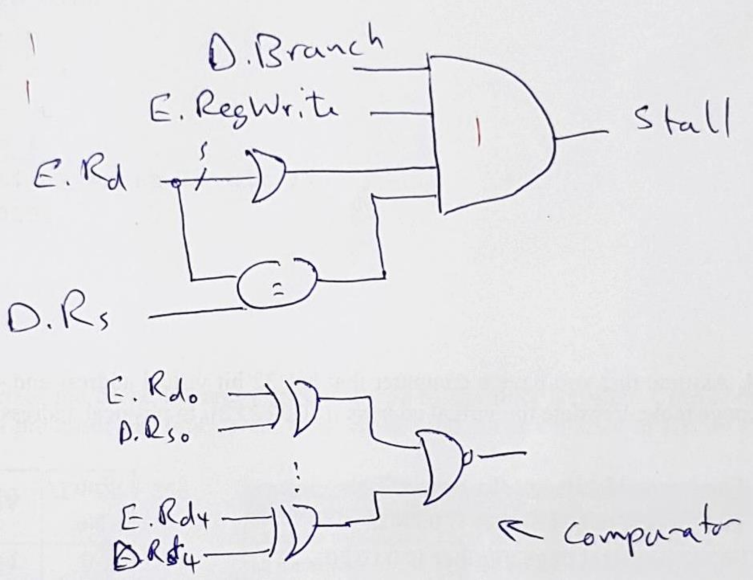
Instructions: Time **100** minutes. Open book and notes exam. No electronics. Please answer all problems in the space provided and limit your answer to the space provided. Every question is for 5 marks. No questions are allowed.

Q1. Assume that you have the classical five-stage pipeline processor that resolves branch instructions in the decode stage. This processor must stall when executing the following sequence of instructions.

```
add $s3, $s1, $s2
beq $s3, $s4, 20
```

Assume that you want to detect this hazard when the branch instruction is in the decode stage. Draw the logic circuit needed to detect this data hazard.

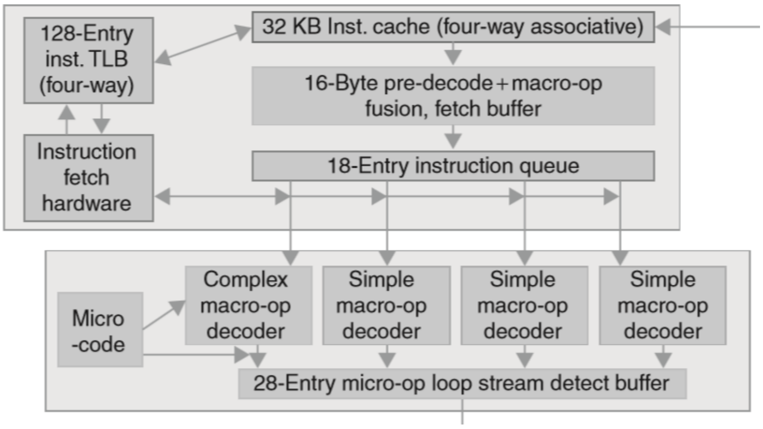
The solution is:



Q2. Assume that the following code sequence is executed by a dynamic dual-issue pipelined processor. This processor uses reservation stations, reorder buffer, and two common data busses to execute the instructions out of order. The fetch stage takes one cycle and the issue stage takes one cycle. The integer latency is 1 cycle, the memory latency is 2 cycles (1 cycle for address calculation and 1 cycle for data memory access), and the floating-point latency is 3 cycles. The processor has one address calculation unit, one memory access unit, one integer ALU unit, and one floating-point unit. Using the multi-cycle pipeline diagram below, specify the execution of these instructions in this processor pipeline.

	1	2	3	4	5	6	7	8	9	10	11	12	13
`l.d  $f2, 0($s1)`	F	I	A	M	W	C							
`l.d  $f4, 0($s2)`	F	I		A	M	W	C						
`add.d $f6, $f4, $f2`		F	I				E	E	E	W	C		
`s.d  $f2, 0($s3)`		F	I		A						C		
`s.d  $f6, 0($s2)`			F	I		A						C	

Q3. Consider the following figure that shows parts of the Intel Core i7 pipeline. What is the main benefit of the macro-op decoders?


128-Entry inst. TLB (four-way)
32 KB Inst. cache (four-way associative)
16-Byte pre-decode + macro-op fusion, fetch buffer
Instruction fetch hardware
18-Entry instruction queue
Micro -code
Complex macro-op decoder
Simple macro-op decoder
Simple macro-op decoder
Simple macro-op decoder
28-Entry micro-op loop stream detect buffer

Solution: They convert complex macro-ops to simple micro-ops that can be more easily pipelined.

Q4. Assume that you have a computer that has 32 bit virtual address and 4-KB pages. Given the following page table, translate the virtual address `00002222`$_{16}$ to physical address.

Entry No.	Valid and Other Bits	Physical Page No.
0	1111 0000 0000	11111$_{16}$
1	1111 0000 0000	21212$_{16}$
2	1111 0000 0000	01010$_{16}$
3	1111 0000 0000	22222$_{16}$
4	1111 0000 0000	33333$_{16}$

The solution is:

The lower 12 bits are the page offset.

The virtual page number is 00002.

So the physical page number is 01010$_{16}$

And the physical address is 01010222$_{16}$

Q5. The following figure shows the parity bits, data bits, and field coverage in a Hamming ECC code for eight data bits. Assume that you have a memory module that uses this ECC code and that the binary sequence `111001000111` is read from the memory and it has a single bit error. What is the original eight data bits?

Bit position		1	2	3	4	5	6	7	8	9	10	11	12
Encoded data bits		p1	p2	d1	p4	d2	d3	d4	p8	d5	d6	d7	d8
Parity bit coverage	p1	X		X		X		X		X		X	
	p2		X	X			X	X			X	X	
	p4				X	X	X	X					X
	p8								X	X	X	X	X

Solution:

123456789012
111001000111
ppdpdddpdddd
The data bits are 1010 0100
p1 = xor(1 1 0 0 0 1) = 1
p2 = xor(1 1 1 0 1 1) = 1
p4 = xor(0 0 1 0 1) = 0
p8 = xor(0 0 1 1 1) = 1
The error code is 1011 -> error in bit 11.
So the data is 1010 0101

Q6. The following table summarizes the cache hierarchy specifications of the Intel Nehalem Core i7-920 processor. Remember that this processor has four cores. What is the total cache capacity in KB on this processor chip?

Characteristic	Intel Nehalem
L1 cache organization	Split instruction and data caches
L1 cache size	32 KiB each for instructions/data per core
L1 cache associativity	4-way (I), 8-way (D) set associative
L1 replacement	Approximated LRU
L1 block size	64 bytes
L1 write policy	Write-back, No-write-allocate
L1 hit time (load-use)	4 clock cycles, pipelined
L2 cache organization	Unified (instruction and data) per core
L2 cache size	256 KiB (0.25 MiB)
L2 cache associativity	8-way set associative
L2 replacement	Approximated LRU
L2 block size	64 bytes
L2 write policy	Write-back, Write-allocate
L2 hit time	10 clock cycles
L3 cache organization	Unified (instruction and data)
L3 cache size	8 MiB, shared
L3 cache associativity	16-way set associative
L3 replacement	Approximated LRU
L3 block size	64 bytes
L3 write policy	Write-back, Write-allocate
L3 hit time	35 clock cycles

Solution:

Size of private caches per core = 32 + 32 + 256
= 320 KB

Total Size = 4 * 320 + 8 * 1024
= 1280 + 8192
= 9472 KB

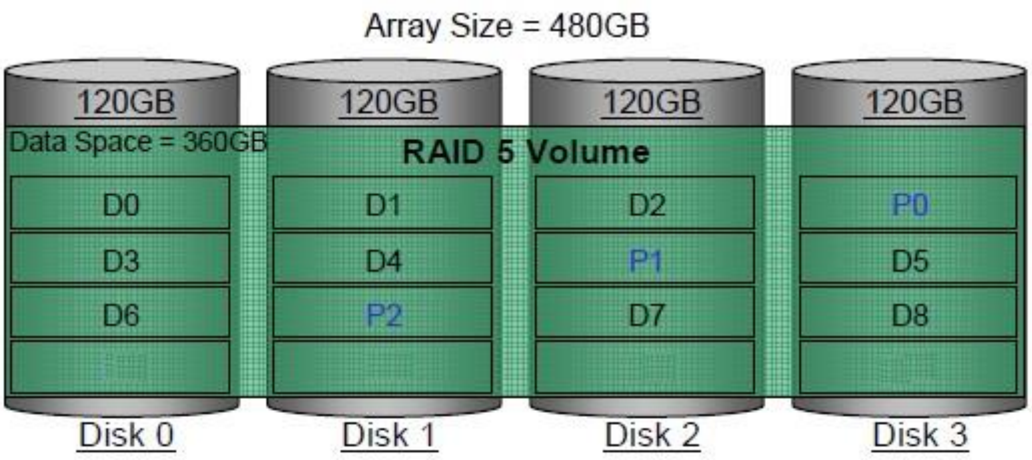
Q7. Assume that you have RAID 5 storage system with a 360-GB data capacity and consisting of a total of 4 disks and configured with 4-KB blocks.

a) What is the raw capacity of each disk?

The solution is: Capacity of one disk = 360 GB / 3 = 120 GB

b) Draw these four disks and show how a 36-KB file is stored in this system.

The solution is:



Q8. Convert the following C code segment to vector MIPS instructions. Assume that `x`, `y`, and `z` are double-precision vectors and their starting addresses are in registers `$s0`, `$s1`, and `$s2`, respectively.

```
for (i=0; i<64; i++)
    z[i] = x[i] * x[i] - y[i];
```

The solution is:

```
lv      $v1,0($s0)
mulv.d  $v2, $v1, $v1
lv      $v3,0($s1)
subv.d  $v4,$v2,$v3
sv      $v4,0($s2)
```

Q9. What is the average memory access time for the memory hierarchy that has the specifications shown in the following table?

Memory Level	**Hit time**	**Miss rate**
L1 cache	2 cycle	10%
L2 cache	10 cycles	2%
L3 cache	20 cycles	1%
Main memory	300 cycles	0%

The solution is:

$\mathbf{AMAT_{L3} = 20 + 0.01 \times 300 = 23\ cycles}$

$\mathbf{AMAT_{L2} = 10 + 0.02 \times 23 = 10.46\ cycles}$

$\mathbf{AMAT_{L1} = 2 + 0.10 \times 10.46 = \underline{3.046\ cycles}}$

Q10. Convert the following C code segment to a shared memory parallel program. Assume that the processor ID is in the variable `Pn`, the number of processors is in the variable `threads`, and `N` is perfectly divisible by `threads`.

```
for (i=0; i<N; i++)
    z[i] = x[i] - y[i];
```

The solution is:

```
for (i=(N/threads)*Pn; i<(N/threads)*(Pn+1); i++)
    z[i] = x[i] - y[i];
```

<Good Luck>