Computer Architecture I Midterm I— (solutions)

Chinese Name: ____________________

Pinyin Name: ____________________

E-Mail ... @shanghaitech.edu.cn: ____________________

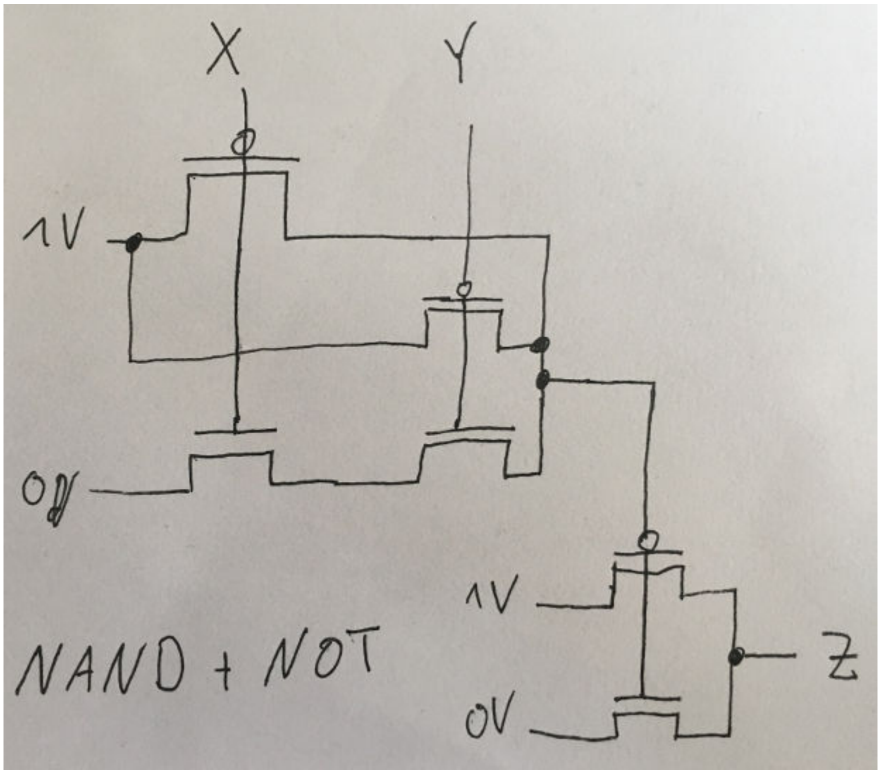
Question	Points	Score
1	1	
2	23	
3	13	
4	18	
5	14	
6	15	
7	16	
8	12	
Total:	112	

- This test contains 15 numbered pages, including the cover page, printed on both sides of the sheet.
- We will use gradescope for grading, so only answers filled in at the obvious places will be used.
- Use the provided blank paper for calculations and then copy your answer here.
- Please turn **off** all cell phones, smartwatches, and other mobile devices. Remove all hats and headphones. Put everything in your backpack. Place your backpacks, laptops and jackets out of reach.
- You have 110 minutes to complete this exam. The exam is closed book; no computers, phones, or calculators are allowed. You may use one A4 page (front and back) of handwritten notes in addition to the provided green sheet.
- The estimated time needed for each of the 6 topics is given in parenthesis. The total estimated time is 95 minutes.
- There may be partial credit for incomplete answers; write as much of the solution as you can. We will deduct points if your solution is far more complicated than necessary. When we provide a blank, please fit your answer within the space provided.
- Do **NOT** start reading the questions/ open the exam until we tell you so!
- Unless otherwise stated, always assume a 32 bit machine for this midterm.

$\boxed{1}$ 1. First Task (worth one point): Fill in you name
Fill in your name and email on the front page and your ShanghaiTech email on top of every page (without @shanghaitech.edu.cn) (so write your email in total 10 times).

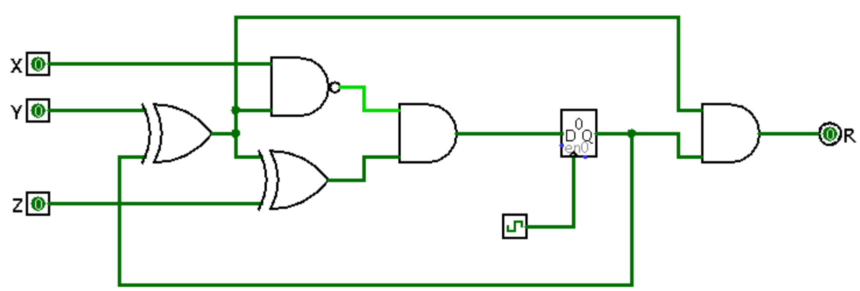
2. SDS (23 points; 20 minutes)

6 (a) Draw a CMOS Network that performs the function AND. Inputs are: X, Y, 1V and 0V, the output is Z. Be sure to make proper use of the special characteristics of the p-channel transistors and the n-channel transistors.



Mystery circuit

You are given the following digital circuit. The register has a setup time of 12ns, a hold time of 7ns, and a CLK-to-Q delay of 8ns, and each logic gate has a delay of 20ns. Assume inputs X, Y and Z are driven by registers with the same specifications.



2 (b) What is the critical path delay, and what is the maximum clock frequency at which the circuit will operate correctly?

Solution: 1x setup (12ns) + 1x CLK-to-Q (8ns) + 3x logic gate (20ns) = 80ns
(The NAND and the XOR below it happen at the same time; the AND at the right side doesn't matter)
1/80ns = 12.5 MHz

8 (c) Assume that during clock cycle 0 the value at the input D of the register is 0. Fill the following table with the values of the register just before the second clock cycle (Q 1), the output R at the same time (R 1) and the output R just before the third clock cycle (R 2), given the input values for X, Y, Z as provided in the table (assume that X, Y and Z keep a constant value at those clock cycles).

Solution: At clock cycle 0 D is 0 => during clock cycle 1 Q is 0 => Q 1 = 0 => R 1 = 0 for all input
For R 2: Q must be 1 => D must have been 1 during cycle 1; Since Q is 1 during cycle 2 Y must be 0 - we need the upper line of the last AND to be 1 - the XOR determining that input is connected to Q (which is 1) so Y must be 0;
So now we need to find a X and Z (given Y=0) that lets D be 1 (given a 0 Q) in cycle 1: The leftmost XOR is 0 (both inputs are 0), so the value of X doesn't matter. So Z must be 1 in order to let the middle AND be 1.
Of course one can also just try to iterate through all input values...

X	Y	Z	Q 1	R 1	R 2
0	0	0	0	0	0
1	0	0	0	0	0
0	1	0	0	0	0
1	1	0	0	0	0
0	0	1	0	0	1
1	0	1	0	0	1
0	1	1	0	0	0
1	1	1	0	0	0

2 (d) Using X, Y and Z as the input and Q 1 as the output, extract a boolean expression using the Sum of Products form from your table.

Solution: This turned out to be an unintentional trick question - it was meant to ask for Q2 (also in the table) and not Q1. The absolute correct answer is:
Q 1 = 0
We will also give full points for any correct sum of products form from your (wrong) values of table 2 (c).

2 (e) Minimize the above form.

Solution: We will accept any answer that is correct given your formula of 2 (d) . Possible answers include "cannot be minimized" or the same formula as 2 (d) if it is already the minimal form.

3 (f) Can you use this minimized from to simply the circuit shown above? Explain how or why not.

Solution: The answer is always NO. This table lacks as input the value Q from the register, so the table and the boolean expression don't represent the complete circuit and can thus not be used to simplify the circuit.

3. Hazardous Conditions (13 points; 15 min)

Assume that we have a standard 5-stage pipelined CPU with **no forwarding**. Register file writes can happen before reads, in the **same clock cycle**. We also have comparator logic that begins at the beginning of the decode stage and **calculates the next PC by the end of the decode stage**. For now, assume there is **no branch delay slot**. The remainder of the questions pertains to the following piece of MIPS code:

	Instructions	Cycle									
		1	2	3	4	5	6	7	8	9	10
0	start: addu $t0 $t1 $t4	IF	D	EX	MEM	WB					
1	addiu $t2 $t0 0		IF	D	EX	MEM	WB				
2	ori $t3 $t2 0xDEAD			IF	D	EX	MEM	WB			
3	beq $t2 $t3 label				IF	D	EX	MEM	WB		
4	addiu $t2 $t3 6					IF	D	EX	MEM	WB	
5	label: addiu $v0 $0 10						IF	D	EX	MEM	WB
6	syscall										

$\boxed{5}$ (a) For each instruction dependency below (the line numbers are given), provide the type of hazard and the length of the stall needed to resolve the hazard. If there is no hazard, write "no hazard".

0 → 1: addu $t0 $t1 $t4 → addiu $t2 $t0 0

0 → 3: addu $t0 $t1 $t4 → beq $t2 $t3 label

1 → 3: addiu $t2 $t0 0 → beq $t2 $t3 label

2 → 3: ori $t3 $t2 0xDEAD → beq $t2 $t3 label

3 → 4: beq $t2 $t3 label → addiu $t2 $t3 6

Solution: 0 → 1: data hazard, 2 cycles
0 → 3: no hazard
1 → 3: data hazard, 1 cycle
2 → 3: data hazard, 2 cycles
3 → 4: control hazard, 1 cycle

For the following questions, assume that our CPU now has forwarding and other optimizations implemented as presented in class and in the book.

$\boxed{4}$ (b) Which of these instruction dependencies would cause a pipelining hazard?

A. ori $t3 $t2 0xDEAD → beq $t2 $t3 label
B. ori $t3 $t2 0xDEAD → addiu $t2 $t3 6
C. ori $t3 $t2 0xDEAD → addiu $t3 $0 10
D. lw $t3 0($t2) → addiu $t4 $t3 4

E. lw $t3 0($t2) → addiu $t4 $t2 4
F. beq $t2 $t3 label → addiu $t2 $t3 6
G. addiu $t2 $t3 6 → beq $t2 $t3 label
H. None of above

(b) ________ **A, D, F, G** ________

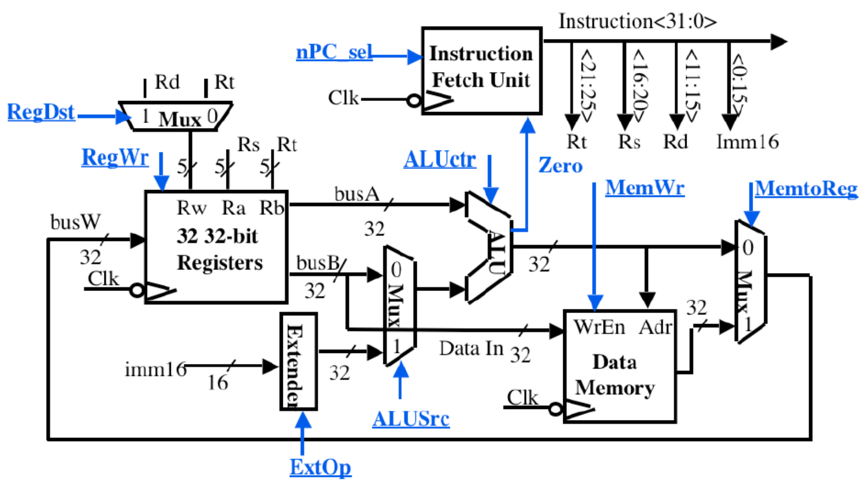
4 (c) If we were given a branch delay slot, which instruction would reduce the most amount of pipelining hazards if moved into the branch delay slot? If all instructions are equally beneficial, or no instruction removes any hazards, write "nop" as your answer.

(c) ________ **5: addiu $v0 $0 10** ________

4. **You have a case of the *soiflz*? Go to Datapathology!** (18 points; 18 min)

See the **single-cycle** MIPS datapath you all know and love below. **Ignore pipelining for the question**. Your job is to modify the CPU to support a new MIPS instruction to perform the following C code in **one** MAL instruction (ptr is an array of int):

```
if(ptr[IMMEDIATE] == 0) {
  ptr[IMMEDIATE] = 1;
}
```



We'll call our new instruction *soiflz*, for store one if load zero. If the word (that is stored IMMEDIATE integers past the base pointer in **rs**) is 0, then set that word to be 1.

5 (a) Make up the syntax for the MAL MIPS instruction that does it (show an example where ptr lives in $t0, and IMMEDIATE is 8). On the right, show the register transfer language (RTL) description of soiflz.

Solution: Syntax:

```
soiflz 8($t0)
```

Syntax	RTL

RTL:

```
if(MEM[R[rs] + SignExtImm] == 0)
   MEM[R[rs] + SignExtImm] = 1;
PC = PC + 4
```

6 (b) Change as *little* as possible in the datapath above **(draw your changes right in the figure)** to enable *soiflz* and list all changes below. Your modification may use adders, shifters, muxes, wires, and new control signals. If necessary, you may replace existing labels. You may not need all boxes.

(i)	WrEn=mux(A=MemWr, B=32 0s|nor32(DataMemoryOut), Control=soiflz)
(ii)	DataMemoryDataIn=mux(A=DataIn, B=0x00000001, Control=soiflz)
(iii)	
(iv)	

5 (c) We now want to set all the control lines appropriately. List what each signal should be: an intuitive name or {0, 1, x − don't care }. Include any new control signals you added.

RegDst	RegWr	nPC_sel	ExtOp	ALUSrc	ALUctr	MemWr	MemtoReg	soiflz		
x	0	+4	Sign	Imm(1)	Add	x	x	1		

2 (d) Your smart friend argues that because of **this very instruction**, you should have a fourth instruction format (in addition to R, I and J). Theres a clear downside: it would cause more complexity with control & datapath. That said, what would be the upside?

Solution: Since you only need 1 register, you could widen the immediate and handle larger offsets.

5. **Memory access** (14 points; 15 min)

Consider a 32-bit physical memory space and a 16 KiB, 5 bits for offset field, 4-way associative cache with LRU replacement. After the execution of the following code:

```
int ARRAY_SIZE = 16 * 1024;
int arr[ARRAY_SIZE]; // *arr is aligned to a cache block
/* loop 1 */
for (int i = 0; i < ARRAY_SIZE; i += 8)
  arr[i] = i;
  arr[i+1] += 1;
/* loop 2 */
for (int i = ARRAY_SIZE - 4; i >= 0; i -= 4)
  arr[i+1] = arr[i];
```

2 (a) 1. We have learnt 6 great ideas in Computer Architecture. So using cache corresponds to which idea ?

(a) **Principle of Locality (Memory Hierarchy)**

2 (b) 2. What is the hit rate of loop 1? What types of misses (of the 3 Cs), if any, occur as a result of loop 1?

(b) **66% hit rate; Compulsory Misses**

2 (c) 3. What is the hit rate of loop 2? What types of misses (of the 3 Cs), if any, occur as a result of loop 2?

(c) **13/16 Capacity Misses**

AMAT

Recall that AMAT = Hit Time + Miss Rate * Miss Penalty.
Suppose your system consists of:
A L1 that hits in 2 cycles and has a local miss rate of 20%
A L2 that hits in 15 cycles and has a global miss rate of 5%
Main memory hits in 100 cycles

2 (d) 1. What is the local miss rate of L2?

(d) **Local miss rate = 5% / 20% = 0.25 = 25%**

2 (e) 2. What is the AMAT of the system?

(e) **AMAT = 2 + 20% x (15 + 25% x 100) = 10**

$\boxed{4}$ (f) 3. Suppose we want to reduce the AMAT of the system to 8 or lower by adding in a L3. If the L3 has a local miss rate of 30%, what is the largest hit time that the L3 can have?

Solution: Let H = hit time of the cache. Using the AMAT equation, we can write: 2 + 20% x (15 + 25% x (H + 30% x 100)) 8 Solving for H, we find that H 30. So the largest hit time is 30 cycles.

6. **IEEE 754** (15 points; 15 min)
Given the indicated number representation, fill in the blank with exactly one of LESS THAN ($<$), GREATER THAN ($>$), or EQUAL TO ($=$). E.g., $1 < 2$ and $1 > 0$.

$\boxed{1}$ (a) IEEE Standard Single-Precision Floating-Point Numbers
A. 1000 0000 0100 1101 1010 0100 1111 0101_{two}
B. 1000 0000 0111 0111 0111 0010 1011 0101_{two}
A ___ B

Solution: A $>$ B (-7.130509E-39 $>$ -1.0969573E-38)
(remember: for IEEE 754 we can compare floating point numbers using just integer comparison - iff we take care of the sign bit; so this can be solved really fast (also for (b) and (c))

$\boxed{1}$ (b) IEEE Standard Single-Precision Floating-Point Numbers
A. 1110 1111 0101 1101 1110 0100 1111 0101_{two}
B. 1110 1100 0111 0111 0111 0000 1011 0101_{two}
A ___ B

Solution: A $<$ B (-6.867298E28 $<$ -1.1965477E27)

$\boxed{1}$ (c) IEEE Standard Single-Precision Floating-Point Numbers
A. 0x9A2F 6AC0
B. 0x9CBC BCBB
A ___ B

Solution: A $>$ B (-3.6275384E-23 $>$ -1.2489582E-21)

16 bit float

We want to develop a new half-precision floating-point standard for 16-bit machines. The basic structure is as follows:

S	Exponent	Significand
15	14 9	8 0

Everything else follows the IEEE standard 754 for floating point, except in 16 bits.

$\boxed{2}$ (d) Originally we will subtract 127 from the exponent, what is the value now?

(d) **31**

$\boxed{2}$ (e) What's the **implicit exponent** when it is a Denormalized number?

(e) **-30**

$\boxed{4}$ (f) Convert the decimal number -10.625 to floating point. Write your answer in hexadecimal.

Solution: Sign:1
Exponent:3->100010
Significant: 010101000
So, 1100 0100 1010 1000
So, 0xC4A8

$\boxed{4}$ (g) What's the **smallest positive number** can represent with those 16 bit?

Solution: $2^{-30} * 2^{-9} = 2^{-39}$

7. Parallel Computation (16 points; 12 min)

8 (a) Name all elements of the Flynn Taxonomy and provide a short description for each element and name an example if you can.

Solution:
Single Instruction Single Data (SISD): one instruction after the other; example: MIPS CPU
Single Instruction Multiple Data (SIMD): one instruction can work on more than one data elements; example: SSE or MMX, AVX
Multiple Instruction Single Data (MISD): multiple (different) instruction working with the same data; no example
Multiple Instruction Multiple Data (MIMD): Different instructions at the same time, with different data; example: multi-core CPUs

1 (b) Provide the formula for Amdahl's Law.

(b) **1 / (1 - F) + F / S)**

4 (c) Assume you parallelize your thesis experimentation program for use on our supercomputer. 90% of the computation can be perfectly parallelized. Initially you are given 36 cores (so the speedup of the parallel part is 36). What is the initial speedup of the program? The due date for your thesis is approaching, so you beg your Prof. for more computation power and are finally given 16384 cores. Calculate the (approximate) speed improvement compared to the initially granted 36 cores.

Solution: Initial: 1/ ((1 - 0.9 + 0.9 / 36) = 1 / (0.1 + 0.025) = 8
16384 cores: 1 / (1 - 0.9 + 0.9 / 16384) approximately 1 / (0.1 + 0) = 10
Improvement: 10/8 = 1.25 == 25% speedup only

3 (d) What is loop unrolling? What are the advantages of it (name at least 3)?

Solution: Take a for loop in C and explicitly write a certain number of those loop iterations in the body of the loop. Advantages:
less loop overhead; it can avoid data hazards; SIMD instructions can be used.

No question here...

8. **Datapath**

Now we want to extend MIPS datapath to accelerate operation **memcpy**, **strcpy**, **memset**. 3 new R-Type instructions, which use counters to quickly scanning through memory values. We use **NL** and **NS** as counters for loads and stores, they both indexed from 0.

rn	Resets both memory counters
lbn \$rd, \$rt, (\$rs)	Computes an address at NL offset from \$rs. The contents of memory at this address are loaded into \$rt. NL is stored into \$rd. Then NL is incremented.
sbn \$rd, \$rt, (\$rs)	Computes an address at NS offset from \$rs. The contents \$rt are stored into memory at this address. NS is stored into \$rd. Then NS is incremented.

We assume that \$rd and \$rt are never equal for the duration of this topic.

4 (a) Fill out the RTL descriptions for these three operations

rn	
lbn \$rd, \$rt, (\$rs)	
sbn \$rd, \$rt, (\$rs)	

4 (b) Fill out the control signals for each of these three instructions hint: for strcpy MIPS code, to copy string at \$a0 to address \$a1

```
strcpy: rn
loop : lbn $0, $t0, ($a0)
     sbn $0, $t0, ($a1)
     bne $t0, $0, loop
     jr $ra
```

	RegDst	RegWr	ALUSrc	ALUCtr	MemWr	MemToReg	NLsel	NSSel	RegWr2	CSel
rn										
lbn										
sbn										

4 (c) Draw control signals added on the graph

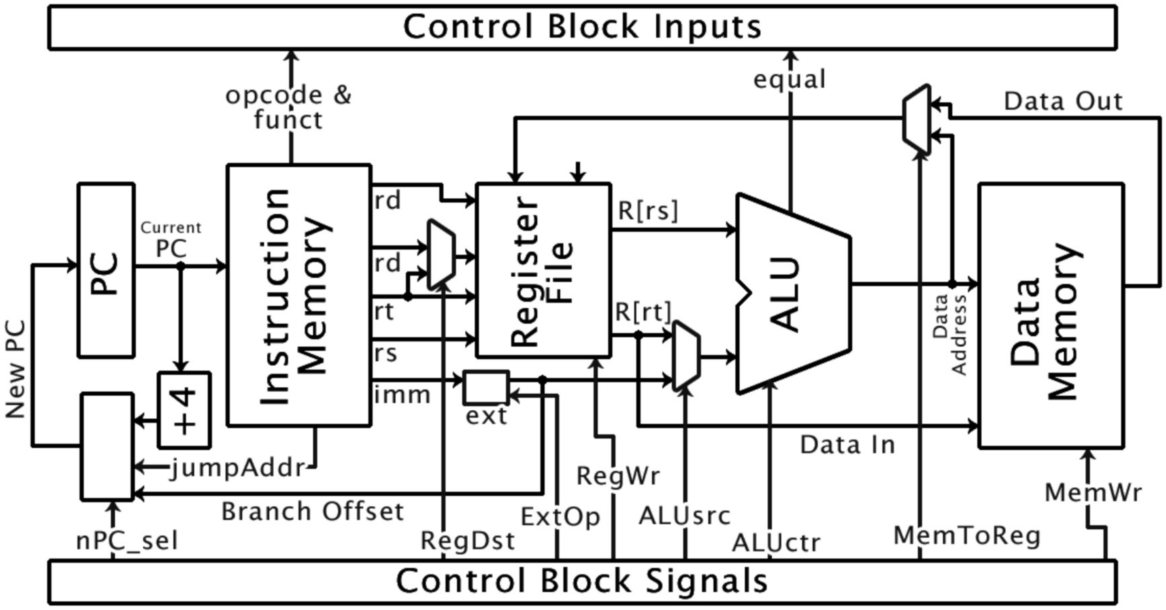
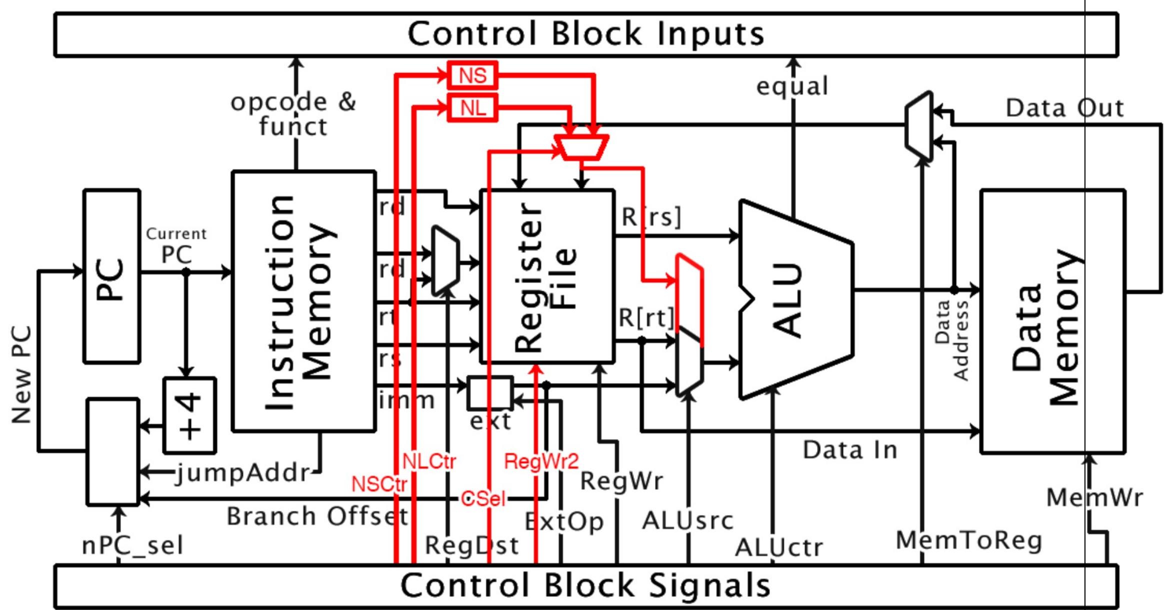


Figure 1: Fill in the control signal

Solution:



	New / Modified Component	New / Modified Control Signal
1	Add NL counter	NLCtr
2	Add NS counter	NSCtr
3	Extend ALUsrc MUX to take an additional input	ALUsrc
4	Add a write enable input for the new register write port	RegWr2
5	Add MUX of NS/NL; wire into new register write port and ALUsrc MUX	CSel

Figure 2: datapath solution