UCLA CCLE

Winter 2021 - COM SCIM151B-1 / EC ENGRM116C-1 - REINMAN

Started on	Wednesday, 3 February 2021, 8:00 AM PST
State	Finished
Completed on	Wednesday, 3 February 2021, 9:29 AM PST
Time taken	1 hour 29 mins
Grade	**61.00** out of 61.00 (**100**%)

Correct

30.00 points out of 30.00

A processor runs a particular application with 5 billion instructions. The processor uses a version of the MIPS ISA, and includes R type instructions (like add, sub, slt), load instructions (LW), store instructions (SW), and branch instructions (BEQ/BNE). The processor does not use the single cycle datapath we have covered in class. Instead, it uses a design where instructions can take multiple cycles to execute, but there is no pipelining: each instruction executes in order, one at a time. The processor has a 2 GHz clock. The instruction latencies and instruction mix for our application are:

	Latency	Instr Mix
R-type	7	45%
LW	11	10%
SW	8	10%
BNE	8	20%
BEQ	8	15%

First - we are going to find the execution time of this application on our processor. For the answers below, round to the nearest hundredth:

Instruction Count (in billions of instructions)=

5

✔

Cycle Time (in nanoseconds)=

0.5

✔

CPI=

7.85

✔

ET=

19.63

✔

We are now going to consider a change to the processor ISA. We frequently see the pattern:

slt REGISTER1, REGISTER2, REGISTER3

bne REGISTER1, $zero, C

Where REGISTER1, REGISTER2, REGISTER3 are any arbitrary registers, $zero is the zero register, and C is any arbitrary constant. And - in this pattern, we know that REGISTER1 is only used by the bne instruction (and no other instruction). To reduce our instruction count, we are going to replace this pattern with a single instruction:

bgt REGISTER3, REGISTER2, C

So REGISTER1 will no longer be written. For example we would replace:

slt $t0, $s0, $s1

bne $t0, $zero, 4096

With:

The bgt instruction will have a latency of 6 cycles. Half of all bne instructions can be found in exactly this pattern. Now we are going to find the new execution time, after recompiling the application to take advantage of this new instruction. For the answers below, round to the nearest hundredth.

New R Instruction Mix (Express as a %) =

38.89

✔ %

New LW Instruction Mix (Express as a %) =

11.11

✔ %

New SW Instruction Mix (Express as a %) =

11.11

✔ %

New BNE Instruction Mix (Express as a %) =

11.11

✔ %

New BEQ Instruction Mix (Express as a %) =

16.67

✔ %

New BGT Instruction Mix (Express as a %) =

11.11

✔ %

Instruction Count (in billions of instructions)=

4.5

✔

Cycle Time (in nanoseconds)=

0.5

✔

CPI=

7.72

✔

ET=

17.37

✔

UCLA CCLE

Correct

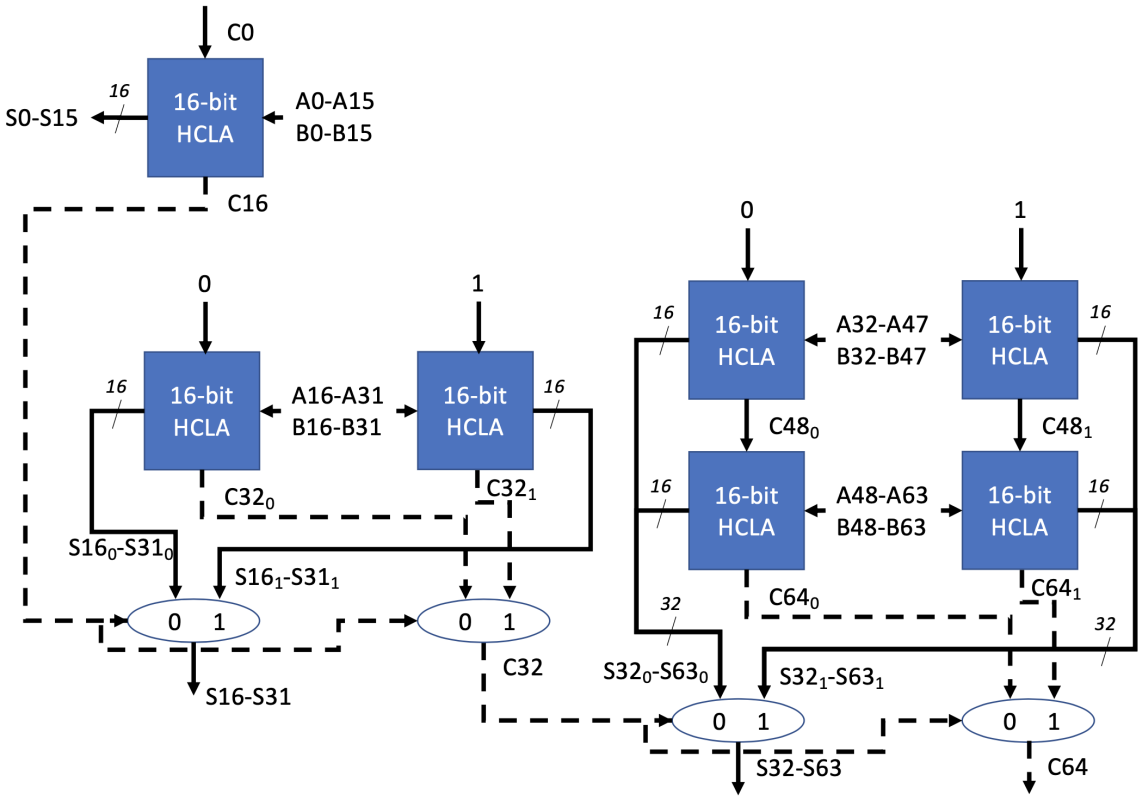
31.00 points out of 31.00

Assume for the rest of this problem that logic gates have the following delays:

Fan-in	AND/OR	XOR
2	2T	4T
3	4T	6T
4	7T	8T
5	9T	10T
6	11T	13T
7	13T	14T
8 or more	15T	15T
Mux	3T	

Where T is some technology dependent parameter. And the Mux delay above is irrespective of fan-in.

Consider the adder design below. It is a 64-bit adder made of seven 16-bit HCLA's that are themselves connected in both carry select and ripple carry fashion:



Each 16-bit HCLA is identical to the design we covered in class (but with the logic gate delays above).

Your task is to find the maximal delay of this design – i.e. determine the delays of $S_{0\text{-}63}$ and C_{64} as labeled on the figure – the maximal delay of these outputs will be the maximal delay of the entire design. To do this (and to help with possible partial credit), fill in the table below.

Assume that the sum logic is implemented with two 2-input XOR gates, just as we did in class.

G0 =

2

P0 =

4

Galpha =

18

Palpha =

11

C16 =

34

G16 =

2

P16 =

4

C32 =

37

C48_0 =

34

C64_0 (i.e. before the mux) =

52

C64 (i.e. after the mux) =

55

C60 =

48

C63 =

62

S63_0 (i.e. before the mux) =

66

S63 (i.e. after the mux) =

69

89

Information

Consider the single-cycle processor implementation from class. Your task will be to augment this datapath with a new instruction – the BiRD – Branch Immediate RD instruction. The BiRD instruction is an I-type instruction. The operation of the BiRD instruction is:

If (R[RT] == SE(RS))

PC = PC + 4 + M[R[$s0]]

Note that BiRD only branches when the register contents of RT matches the sign extended value of RS. Note that RS is not being used as a register specifier here - it is being used as a 5-bit immediate. When branching, the PC displacement is the memory contents of the address in register $s0 - register $so is always used with this instruction - it is an implicit operand.

Implement your solution on the datapath and control tables from class.

◄ Midterm Exam Work ...

Jump to...

Q3 grading guidelines ►