CSM 61C Fall 2020

CALL and SDS

Mentoring 4: October 5, 2020

1 CALL

CALL is an acronym for the process that C source code goes through before being executed by the computer. The four steps are Compiler, Assembler, Linker, and Loader. One benefit of using so many stages is that individual parts of a program can be compiled separately before being linked into the final executable. This can be very helpful when working with massive programs!

	Input	Output
Compiler	C source code (foo.c) #directives have already been handled by the preprocessor before this step	Assembly language (foo.s for RISC-V) Output can contain pseudo instructions like la, nop, li, mv, j, etc.
Assembler	Assembly language (foo.s)	Object file containing object code & information tables (foo.o for RISC-V) Directives (.text, .word, .data, etc.) which specify formatting without producing any machine code are interpreted and pseudo instructions are replaced. Symbol table, which lists functions and variables that can be called by other files, and relocation table, which tracks labels and data within the code that still needs to be filled in, are created.
Linker	One or more object files (foo.o, lib.o, stuff.o) Each object file can be compiled up to this point independently, so an update to a library or a refactoring of code does not require recompilation of everything.	Executable (a.out) The text segments from each file are concatenated together and the data segments likewise. The reference tables are resolved as the linker has access to the final text/data section and can calculate absolute addresses/offsets.
Loader	Executable (a.out) Handled by the OS	Program execution by computer Header is read and instructions/data from the executable are copied into a newly allocated address space

1.1 The following are some multiple choice questions about CALL. Clearly circle the correct answer:

(a) A system program that combines separately compiled modules of a program into a form suitable for execution is _____.

A. Assembler
B. Loader
C. Linker
D. None of the Above

C

(b) At which point will all the machine code bits be determined for a *la* instruction?

A. C code
B. Assembly code
C. Object code
D. Executable

D

(c) At the end of the assembling stage, the symbol table contains the _____ of each symbol.

A. relative address
B. absolute address
C. the stack segment beginning address
D. the global segment beginning address

A

(d) beq and bne instructions produce _____ and they _____.

A. PC-relative addressing, never relocate
B. PC-relative addressing, always relocate
C. Absolute addressing, never relocate
D. Absolute addressing, always relocate

A

(e) jal and jalr instructions add symbols and _____ to _____.

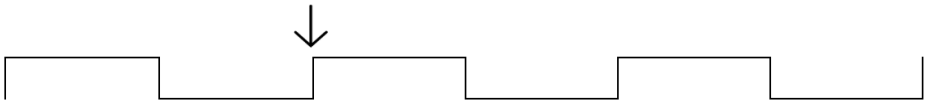
A. instruction addresses, the symbol table
B. symbol addresses, the symbol table
C. instruction addresses, the relocation table
D. symbol addresses, the relocation table

C

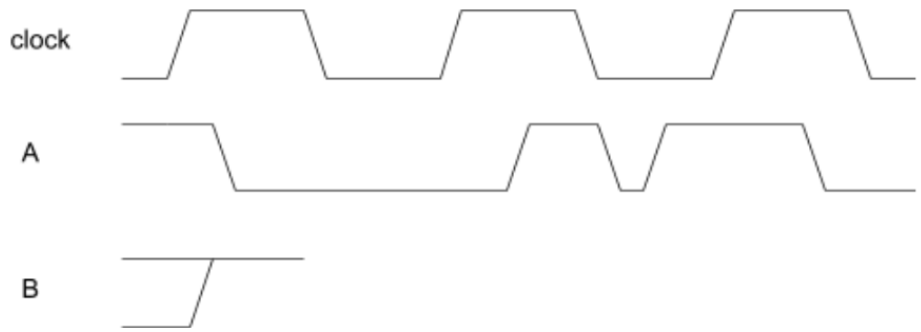
2 Synchronous Digital Systems

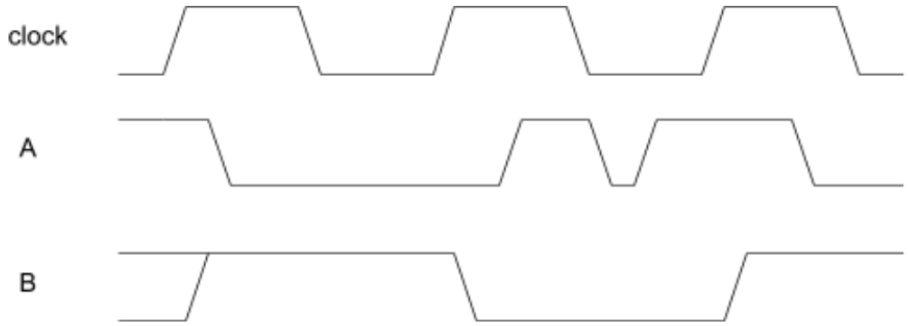
Synchronous Digital Systems (SDSs) are systems designed to process input as it comes in and output it as quickly as it comes in. The input is always a series of pins with either high or low voltage, and the output is another series of pins, each with either high or low voltage. A clock is used to tell the input and output when to switch to the next value; the switch always occurs on the rising edge (low to high voltage) of the clock. This is so that the output is always consistent, otherwise halfway through the switch of the input, the output might be based on the old input or the new, depending on the quality of the circuit and also some random chance based on electron movement. Note: the output always takes a small amount of time to stabilize after the rising edge.

When slowed down, clock cycles will look something like below. This is three clock cycles in a row, and here is the rising edge:



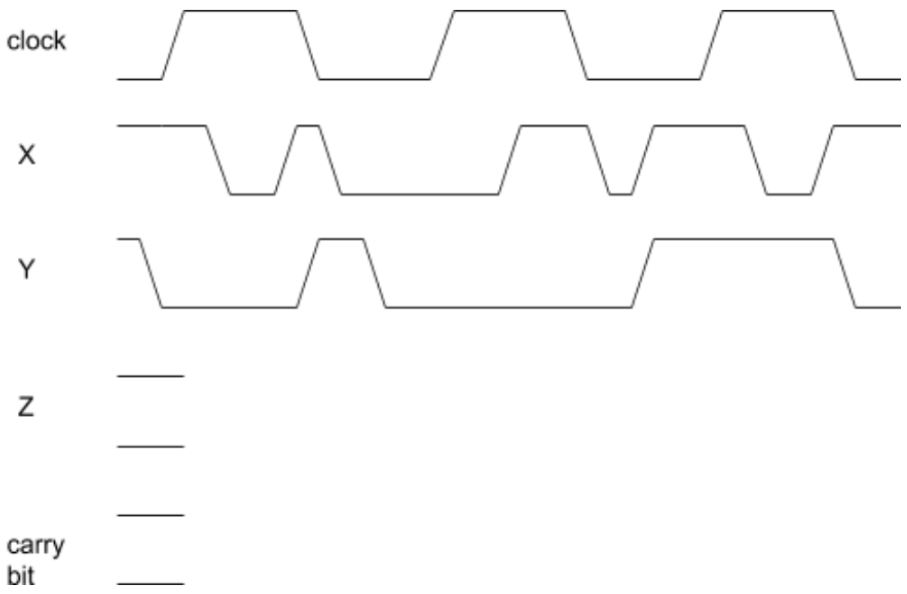
2.1 On the rising edge of the clock, B is set to A and is held constant until the next rising edge. Fill in the diagram for B's value.

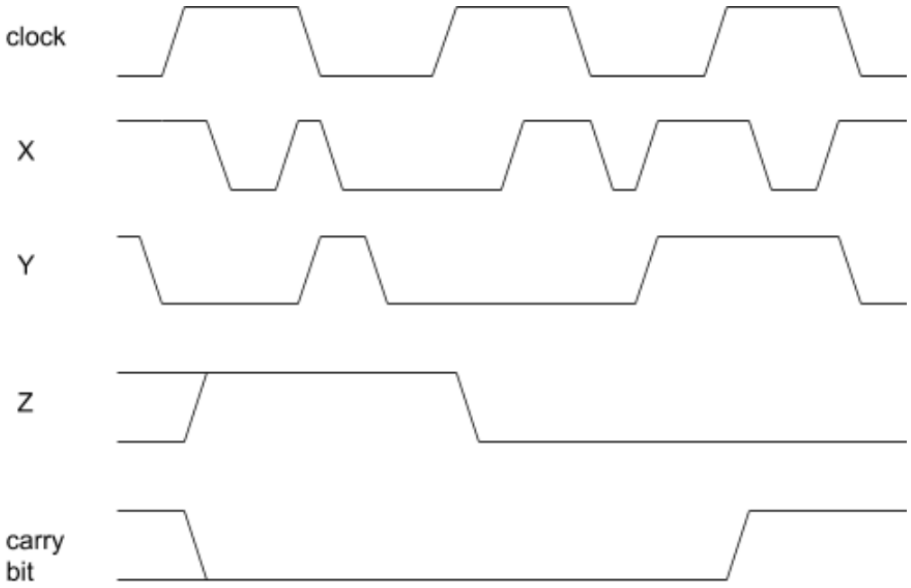




Solution:

2.2 In a clock cycle, complex operations can occur. For example, most of the RISC-V instructions can be done in 1 clock cycle. Below, the diagram shows the equation Z = X + Y where Z is computed on the rising edge of the clock. If Z overflows, place the overflow in the carry bit.





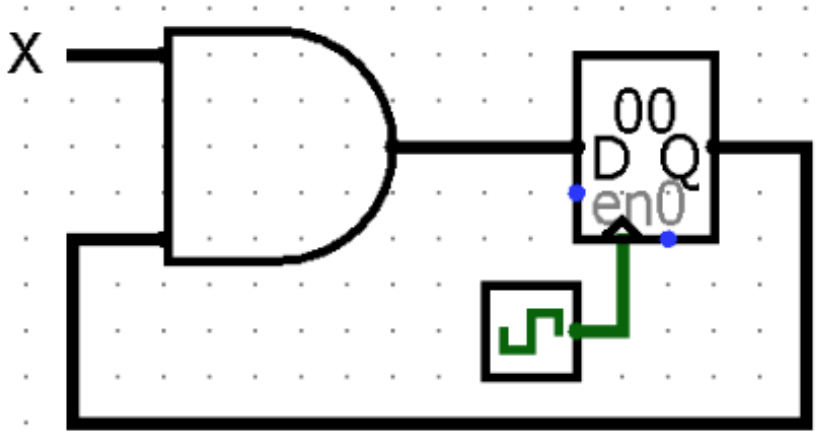
Solution:

3 SDS With Registers

For the circuit below, assume:

1. setup time is 15ns
2. hold time is 30ns
3. AND gate delay is 10ns.

If the clock rate is 10 MHz and x updates 25ns after the rising edge of the clock,what are the minimum and maximum values for the clk-to-Q delay to ensure proper functionality?



3.1 Min: _____ ns

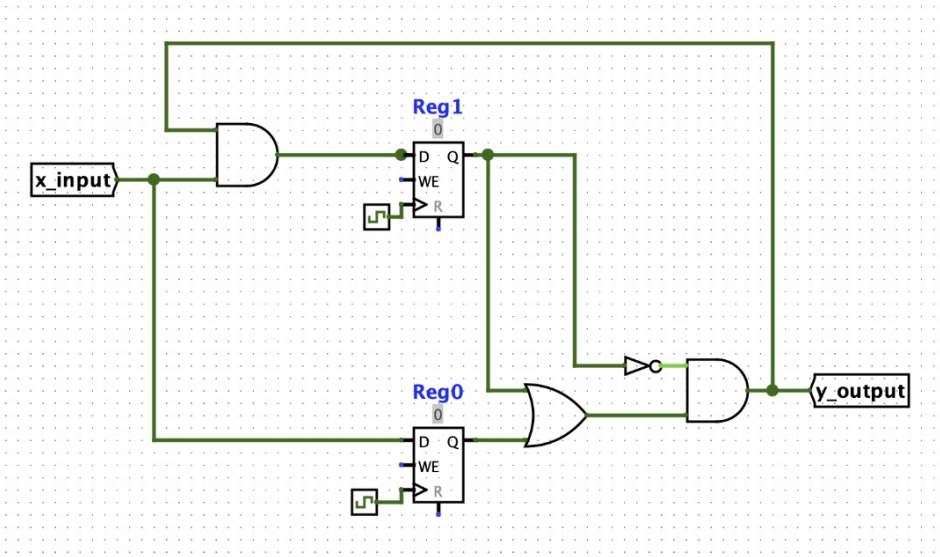
3.2 Max: _____ ns

Min: 20 ns

Max: 75 ns

If the clk-to-Q delay is too fast, the input to the register will change before the hold time is finished. Thus, the minimum clk-to-Q delay is t_{hold}–t_{and} = 30–10 = 20 ns. On the other hand, we must make sure the critical path is no longer than the clock period, which is $100ns (= 1/(10MHz))$. In other words, $t_{setup}+t_{and}+t_{clk-to-Q} \leq 100ns$, or $tclk-to-Q \leq 100ns-t_{setup}-t_{and}$. Solving yields $t_{clk-to-Q} \leq 75ns$.

4 Logic, SDS (Summer 2020 Final Q4)



4.1 For this question, assume:

- **AND** gates have a propagation delay of 9ns
- **OR** gates have a propagation delay of 14ns
- **NOT** gates have a propagation delay of 5ns
- **x_input** switches value 30ns after the rising edge of the clock
- **y_output** is directly attached to a register
- **Setup** time is 3ns
- **Clk-to-Q** delay is 4ns

Given the above circuit diagram, answer the following questions:

(a) What is the max hold time in ns?

18 ns Shortest CL between registers: NOT + AND = 5 + 9 = 14ns
Max hold time = Clk-to-Q + Shortest CL = 4 + 14 = 18ns

(b) What is the minimum clock period in ns?

42 ns Critical path = 30ns (for x_input to change) + 9 (AND) + 3 (Setup) = 42ns

(c) Regardless of your previous answers, assume the clock period is 50ns, the first rising edge of the clock is at 25ns and x_input is initialized to 0 at 0ns. At what time in ns will y_output become 1?

102 ns 25 (First rising edge) + 50 (clock period) + 4 (Clk-to-Q) + 14 (OR) + 9 (AND) = 102ns

(d) How long will y_output remain equal to 1 before switching to 0?

50ns If y_output changes to 1 at 102, the next rising edge is at 125. From there, it takes Clk-Q (4) +OR (14) + AND (9) = 27 ns to update y_output to 0 again, so at 125 + 18 = 152 ns, 152 - 102 = 50ns