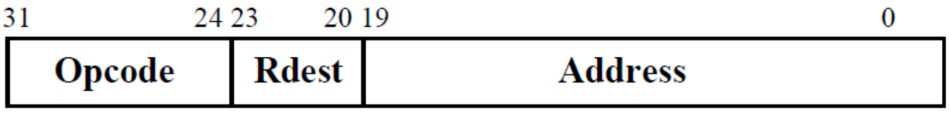
112 Intro to Computer System 2020 Exam Sample Solution

Disclaimer: This is not an official answer key, so please correct any mistake if you find one. There are more than one possible solution for some questions, so keep in mind about that!

1. **a.** We can realize a one-bit memory using an SR-latch flip or D-type latch flipflop(this ons is better). We need an adress decoder to access several bits memory.

   RAM is a sequential circuit because RAM output depends on the past inputs(the previous write operations) which makes RAM stateful and needs to be implemented using sequential logic. However, if we only consider read operations in RAM, excluding the write operation, static RAM is combinational because reading only involces decoding the address and put the bits in the memory on the output circuit. For dynamic RAMs, they are sequential since they have to complete their refresh cycle before read/writes.

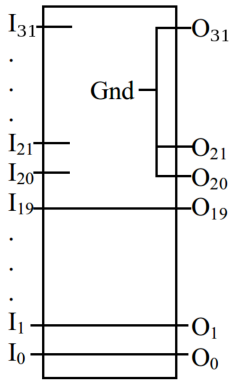
   b. The instruction format looks like this:



$$\text{Load: lw Rdest, Addr}$$
$$\text{Store: sv Rdest, Addr}$$
$$\text{Jump: jmp Addr (Ignore Rdest field)}$$
$$\text{Call: call Rdest, Addr}$$

(The naming of the abbreiviation of those instructions do not matter)

The bit mask circuit looks like this:



This circuit does not need any gates. The design has a drawback: it would restrict the size of the data/address field carried by the instruction to $2^{20} = 1\text{M}$

c. The instruction execution is described as follows:

Instruction	Cycle	Transfers	Path
LOAD Rdest, Address	E1	MAR←MDR	Use the bit mask
	E2	MDR←Memory	
	E3	Rdest←MDR	No masking
STORE Rdest, Address	E1	MAR←MDR; A←Rdest	Use the bit mask
	E2	Memory←A	via the Shifter (no change)
JUMP Address	E1	PC←MDR	Use the bit mask
CALL Rdest, Address	E1	PC←PC+1	
	E2	Rdest←PC	
	E3	PC←MDR	Use the bit mask

d. The instruction execution is described as follows:

Instruction	Cycle	Transfers	Path
LOADINDIRCET Rdest, Rscr	E1	A←Rsrc	
	E2	MAR←A	
	E3	MDR←Memory	
	E4	Rdest←MDR	No masking
STOREINDIRECT Rdest, Rscr	E1	A←Rscr	
	E2	MAR←A;A←Rdest	via the shifter (no change)
	E3	Memory←A	via the shifter (no change)
JUMPINDIRECT Rscr	E1	A←Rscr	
	E2	PC←A	via the shifter (no change)
CALLINDIRECT Rdest, Rscr	E1	PC←PC+1;A←Rscr	
	E2	Rdest←PC;	
	E3	PC←A	via the shifter (no change)

e. Possible solutions:

- Reduce the size of instrucitons for certain operations such as add and substract, which would only use 16 bits instead of full 32 bits. This can reduce the number of fetch cycles.
- Implement fast addition/multiplier hardwares to process several bits in parallel.
- Optinmze the combination logic so that each clock cycle has less delay and becomes faster.
- Design a better controlling logic for the controller of CPU and memory.
- Use faster memory/cache (but could be more expensive) or optimize the instruction set (i.e. more uniform and reduce nunber of cycles needed for register transfer operations).

2. **a.** A possible simplification:

$$
\begin{aligned}
E &= A' \cdot (A + B) + (B + A \cdot A) \cdot (A + B') \\
&= A' \cdot (A + B) + (B + A) \cdot (A + B') && \text{(By absorption)} \\
&= A' \cdot A + A' \cdot B + B \cdot A + B \cdot B' + A \cdot A + A \cdot B' && \text{(By distributivity)} \\
&= 0 + A' \cdot B + B \cdot A + 0 + A + A \cdot B' && \text{(By } A \cdot A' = 0 \text{ and } A \cdot A = A\text{)} \\
&= B \cdot (A' + A) + A \cdot (1 + B') && \text{(By Inverse of distributivity)} \\
&= B + A && \text{(By } A' + A = 1 \text{ and } 1 + A = 1\text{)}
\end{aligned}
$$

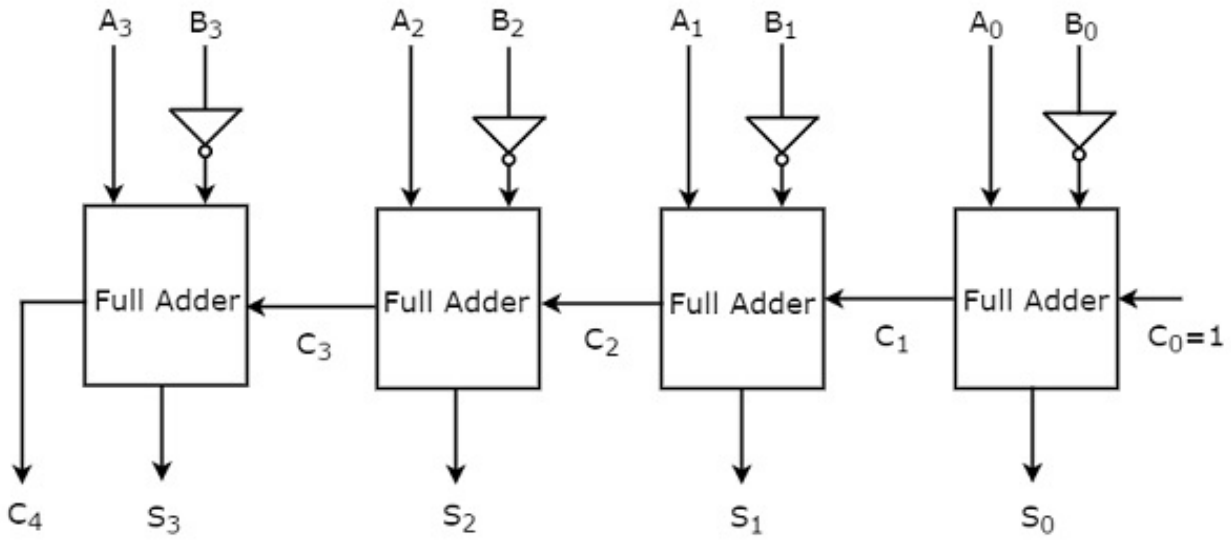
b. i) $C_{out} = A' \cdot B \cdot C_{in} + A \cdot B' \cdot C_{in} + A \cdot B \cdot C'_{in} + A \cdot B \cdot C_{in}$

$$S = A' \cdot B' \cdot C_{in} + A' \cdot B \cdot C'_{in} + A \cdot B' \cdot C'_{in} + A \cdot B \cdot C_{in}$$

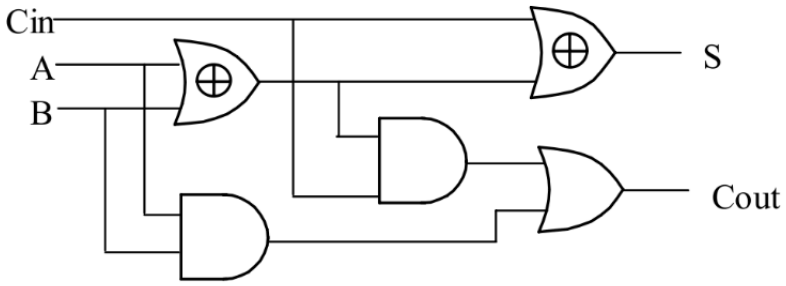
ii) $C_{out} = (A + B + C_{in}) \cdot (A + B + C'_{in}) \cdot (A + B' + C_{in}) \cdot (A' + B + C_{in})$

$S = (A + B + C_{in}) \cdot (A + B' + C'_{in}) \cdot (A' + B + C'_{in}) \cdot (A' + B' + C_{in})$

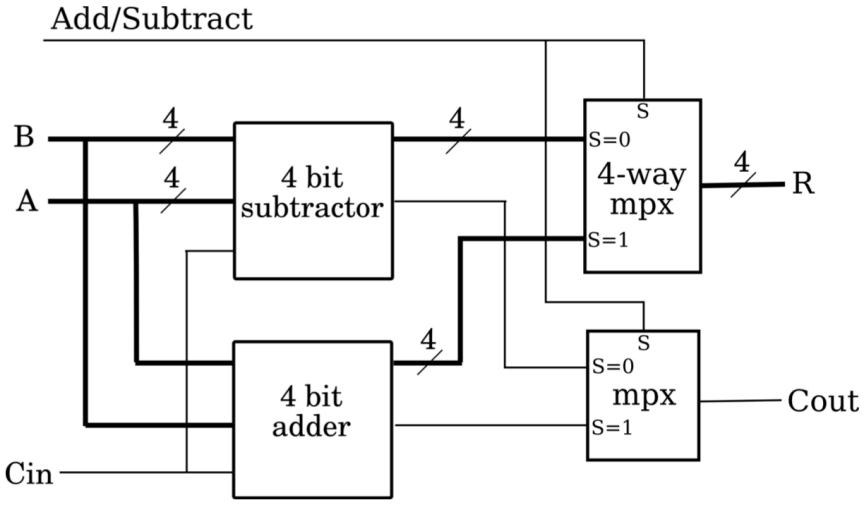
c. Below is a 4-bits two's complement subtractor:



Where each box represents a full adder:



d. A possible implementation:



e. -$139.325 = 1\ \ 10000110\ \ 00010110101000110011001 1_{\text{bin}}$=C30B 5333_{hex}

Sign bit: Negative means that the sign bit is 1.

Biased exponent and Mantissa: $139 = 10001011_{\text{bin}}$

$$0.325 = 0101001100110011_{\text{hex}}$$

The calculation of 0.325 is below:

$$\begin{aligned}
0.325 * 2 &= 0.65 \\
0.65 * 2 &= 1.3 && (\text{Bigger than 1, so we minus 1}) \\
0.3 * 2 &= 0.6 \\
0.6 * 2 &= 1.2 && (\text{Bigger than 1, so we minus 1}) \\
0.2 * 2 &= 0.4 \\
0.4 * 2 &= 0.8 \\
0.8 &= 1.6 && (\text{Bigger than 1, so we minus 1}) \\
0.6 * 2 &= 1.2 && (\text{Begining here, it will be repeating})
\end{aligned}$$

Collecting the unit digits we have $0.325 = 0101001100110011.....$ (0011 is the repeating sequence).

Hence,
$139.325 = 10001011.0101001100110011... = 1.00010110101001100110011... \times 2^7$.

The real exponent is 7, hence the biased exponent is $7 + 127 = 134 = 10000110$.

The Mantissa is 00010110101001100110011 which we only preserve 23 bits because of single precision.

Hence we have the resuit: $1 \;\; 10000110 \;\; 00010110101001100110011$

It's hexadecimal equivalent is C30B 5333