Tell us what the following numbers mean if interpreted as (i) 2's comp (ii) Unsigned.

a. 0x20080024

b. 0xAC4F0004

20080024 is positive in 2's comp, so same as unsigned:

$2 \times 16^7 + 8 \times 16^4 + 2 \times 16^1 + 4 = 537395236_2$

AC4F0004 unsigned

$10 \times 16^7 + 12 \times 16^6 + 4 \times 16^5 + 15 \times 16^4 + 4 = 2890858500$

2's comp

A C 4 F 0 0 0 4

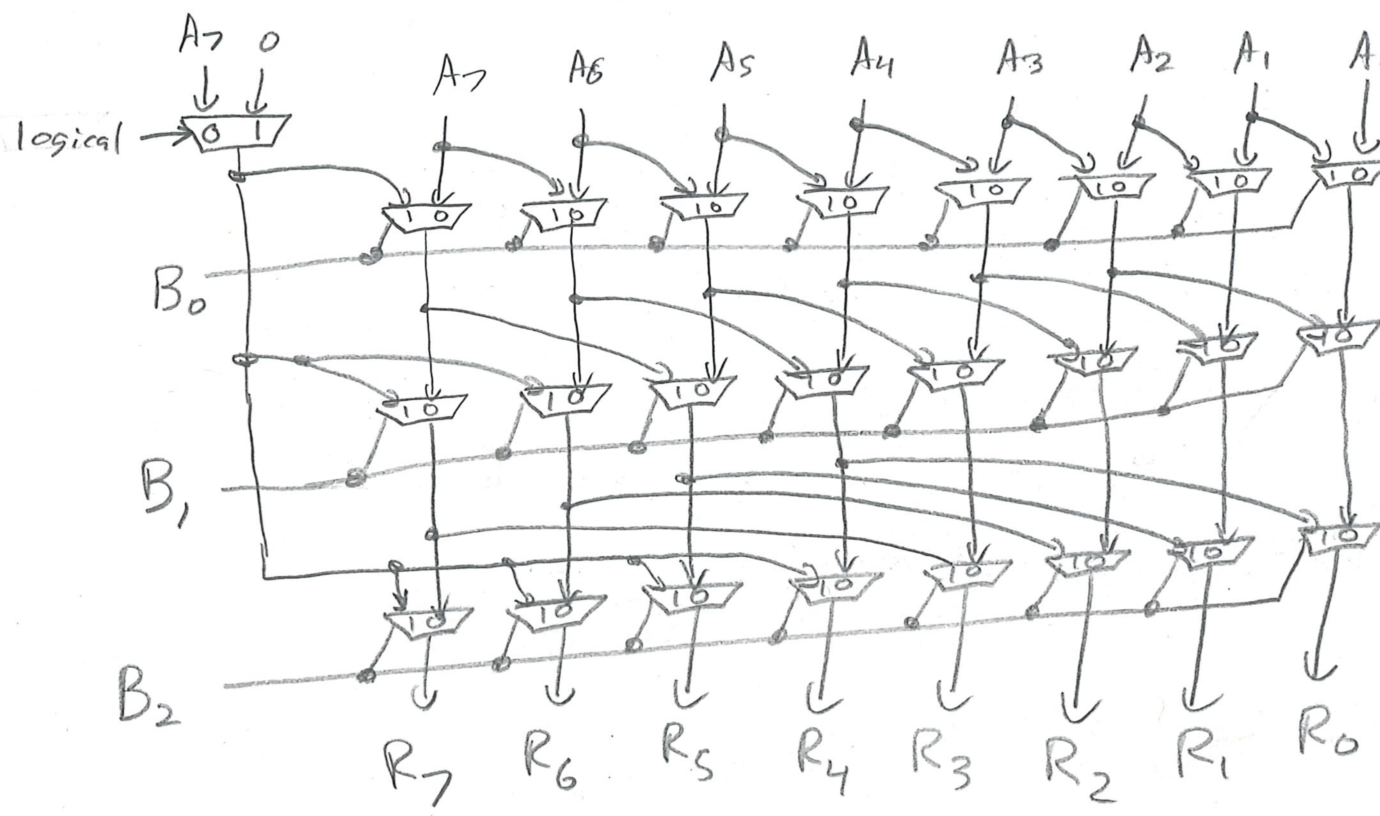
$= --(1010\ 1100\ 0100\ 1111\ 0000\ 0000\ 0000\ 0100)$

$= -(0101\ 0011\ 1011\ 0000\ 1111\ 1111\ 1111\ 1011\ +1)$

$= -\ 5\ 3\ B\ 0\ F\ F\ F\ C$

$= -1404108796$

In class we build a left shifter. Using the same logic, create a right shifter for 8-bit values. It takes in an 8-bit value $A_7..A_0$ to be shifted, and a 3-bit shift amount $B_2..B_0$. It also gets a signal "Logical", which when true performs a logical right shift, while when false performs an arithmetic shift (same as division by 2^B on 2's comp numbers). Your design should require only 2:1 Muxes, and should be as simple as possible.



Assuming the values given below were unsigned 6-bit values, compute the following results. Your answers should be a 12-bit unsigned value

A: 100101

B: 001011

C: 110010

i.) A * B

ii.) A * C

```
i.)            100101
             x 001011
             --------     x1
               100101     x2
              100101-     x0
             000000--     x8
            100101---     x0
           000000----     x0
       + 000000-----
       -----------------
         00110010111
```

```
ii.)           100101
             x 110010
             ---------
               000000     x0
              100101-     x2
             000000--     x0
            000000---     x0
           000000----     x16
          100101----      x32
     +   100101-----
     -------------------
         11100111010
```

Suppose you wish to run a program P with 7.5×10^9 instructions on a 5GHz machine with a CPI of 0.8.

a.) What is the expected CPU time?

b.) When you run P, it takes 3 seconds of wall clock time to complete. What is the percentage of the CPU time P received?

a.) $\text{Exec Time} = \text{instr} \times \text{CPI} \times \frac{1}{\text{Rate}}$

$$7.5 \times 10^9 \times 0.8 \times \frac{1}{5 \times 10^9}$$

$$= 1.5 \times 0.8 = 1.2 \text{ Seconds}$$

b.) $\frac{1.2}{3} = 40\%$

Consider program P, which runs on a 1 GHz machine M in 10 seconds. An optimization is made to P, replacing all instances of multiplying a value by 4 (mult X, X, 4) with two instructions that set x to x+x twice (add x, x, x; add x, x, x). Call this new optimized program P'. The CPI of a multiply instruction is 4, and the CPI of an add is 1. After recompiling, the program now runs in 9 seconds on machine M. How many multiplies were replaced by the new compiler?

OLD: $\text{Exec} = \text{instr} \times \text{CPI} \times \frac{1}{\text{Rate}}$

OLD: $10 = (\text{Norm\_instr} \times \text{Norm\_CPI} + \text{mult4} \times 4) \times \frac{1}{1 \times 10^9}$

$10 \times 10^9 = (\text{Norm\_instr} \times \text{Norm\_CPI} + 4 \times \text{mult4})$

NEW: $9 = (\text{Norm\_instr} \times \text{Norm\_CPI} + (2 \times \text{mult4}) \times 1) \times \frac{1}{1 \times 10^9}$

$9 \times 10^9 = (\text{Norm\_instr} \times \text{Norm\_CPI} + 2 \times \text{mult4}$

OLD - NEW: $10 \times 10^9 - 9 \times 10^9 = (\text{Norm\_instr} \times \text{Norm\_CPI} + 4 \times \text{mult4})$
$- (\text{Norm\_instr} \times \text{Norm\_CPI} + 2 \times \text{mult4}$

$1 \times 10^9 = 4 \times \text{mult4} - 2 \times \text{mult4}$

$1 \times 10^9 = 2 \times \text{mult4}$

$\underline{\underline{0.5 \times 10^9}} = \text{mult4}$

Your company could speed up a Java program on their new computer by adding hardware support for garbage collection. Garbage collection currently comprises 20% of the cycles of the program. You have two possible changes to the machine. The first one would be to automatically handle garbage collection in hardware. This causes an increase in cycle time by a factor of 1.2. The second would be to provide for new hardware instructions to be added to the ISA that could be used during garbage collection. This would halve the number of instructions needed for garbage collection but increase the cycle time by 1.1. Which of the two options, if either, should you choose?

Baseline: Exec Time = Original

HW : ExecTime = 80% × Original × 1.2 = .96 × Original

GCinstr : Exec Time = 1.1 × (.8 × Original + .2 × Original / 2)

= 1.1 × (.8 + .1) × Original

= 1.1 × 0.9 × Original

= .99 × Original

HW is best