ECE 1175 Midterm solution

1.a

- **Requirements**

  Tell people what is the goal and what they expect to get.
- **Specification**

  Provide inputs to the architecture design process.
- **Architecture**

  The major components that can satisfy the specifications.
- **Component design**

  Design hardware and software components.
- **System integration**

  Put components together.

1.b

(1) A biomedical sensor

- **Size & weight**

  The sensor should be tiny and lightweight such that it won't affect the patient's daily activities.
- **Power consumption**

  The sensor should be low-power to achieve extended period of monitoring.
- **Accuracy**

  The measurement collected by the sensor should be highly accurate to reflect the actual heart rate of the patient.

(2) Sensor motes for noise monitoring

- **Durability**

  The sensor motes should keep alive for months or years by harvesting solar energy and being able to withstand bad weather.
- **Accuracy**

  The sensor should collect reliable and high-resolution readings for later analysis.
- **Deployment cost**

  The sensor should be cheap to be massively deployed throughout the city.

(3) Parking distance control sensors

- **Accuracy**

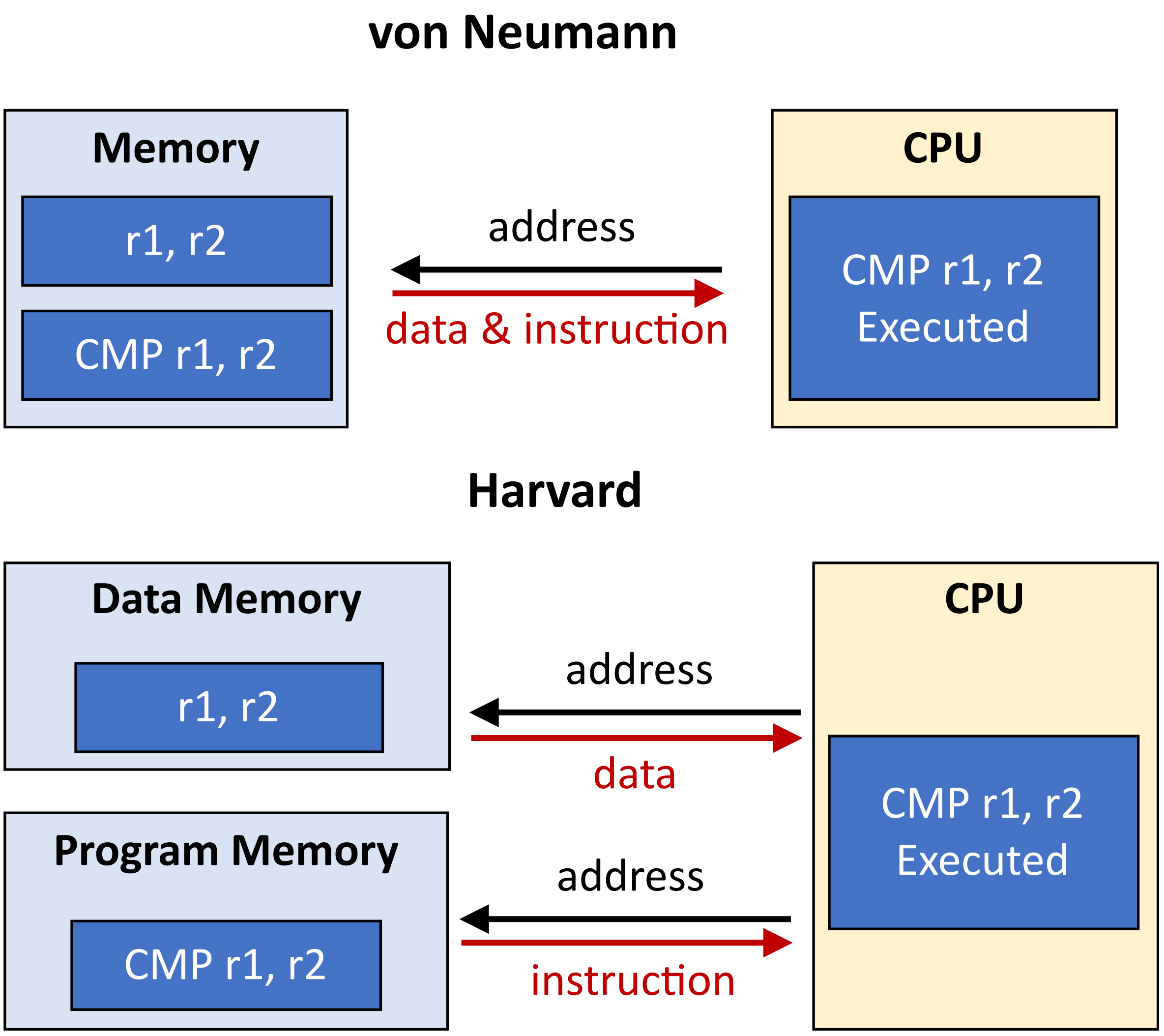
  The sensor measurement should be extremely accurate to avoid misleading the drivers.
- **Power consumption**

The sensor should be low-power to keep alive for months or years. Unexpected power outage will cause damage.

- **Latency**

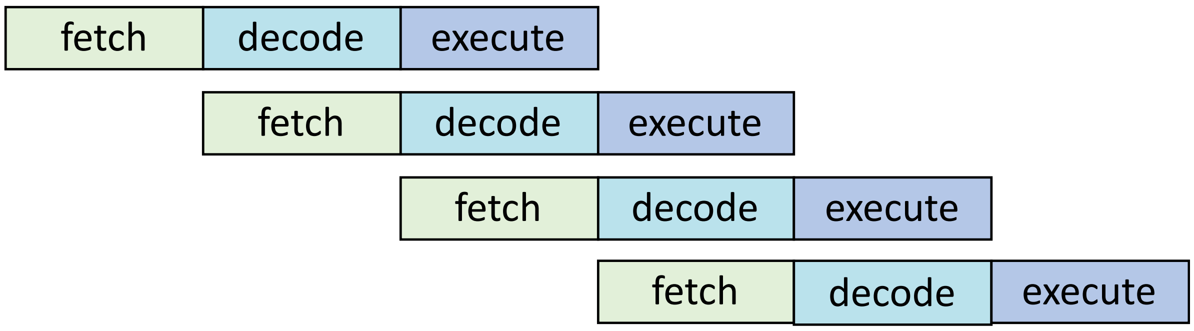
  The sensor should have very low response time to provide timely distance status.

2.a



2.b

RSIC adopts compact and uniform instruction format to facilitate pipelining. Multiple instructions are overlapped during execution.



2.c

```
MOV r0, #0      ; i = 0
MOV r1, #0      ; x
loop:
CMP r0, #5
BGE endloop     ; stop loop if i >= 5
CMP r0, #4
MOVEQ r1, #0    ; x = 0 if i == 4
MOVNE r1, #1    ; we can safely do this because the previous
    ; MOV doesn't affect status bits
ADD r0, r0, #1  ; i++
B loop
endloop:
```

3.a

```
// attempt lock, and test if success or not
TestAndSet(){
    oldValue = peek(LOCK);
    poke(LOCK, true);
    return oldValue;
}

// wait until lock is acquired
while(TestAndSet());

// write to the output register
poke(OUT_CHAR, *current_char);
// wait for device to finish by checking its status
while(!peek(OUT_STATUS));
// proceed to the next character
current_char++;
// release lock after write is finished
poke(LOCK, false);
```

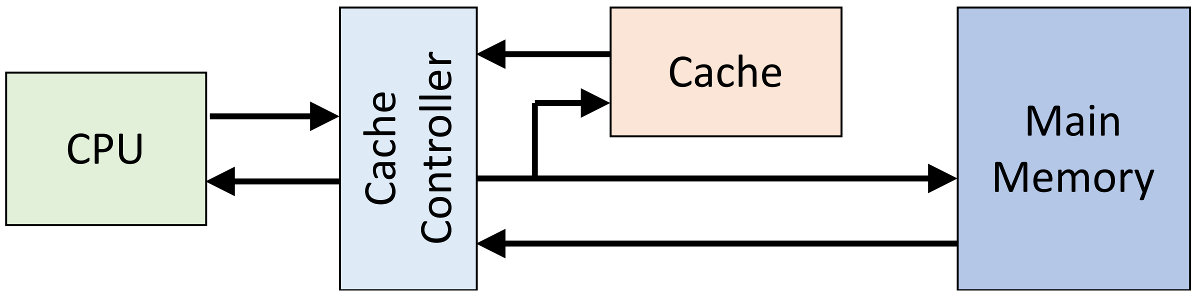
3.b

The system needs to handle interrupts from multiple devices. Some interrupts are more urgent to be handled.

Interrupt with priority lower than current priority is not recognized until current interrupt is complete.

Example: Someone presses Ctrl+C to kill a program which is endlessly generating interrupt.

4.a



4.b

$$\bar{t} = h_1 t_1 + (1 - h_1)(h_2 t_2 + (1 - h_2) t_{\text{main}}) = 13.4 \text{ ns}$$

5.a

Each address splits into row and column address corresponding to the 2-D array.

For example, say a 16x16 2-D array stores the data with address from 0x00 to 0xFF. Then the address 0x64 corresponds to (row=6, col=4) of the array.

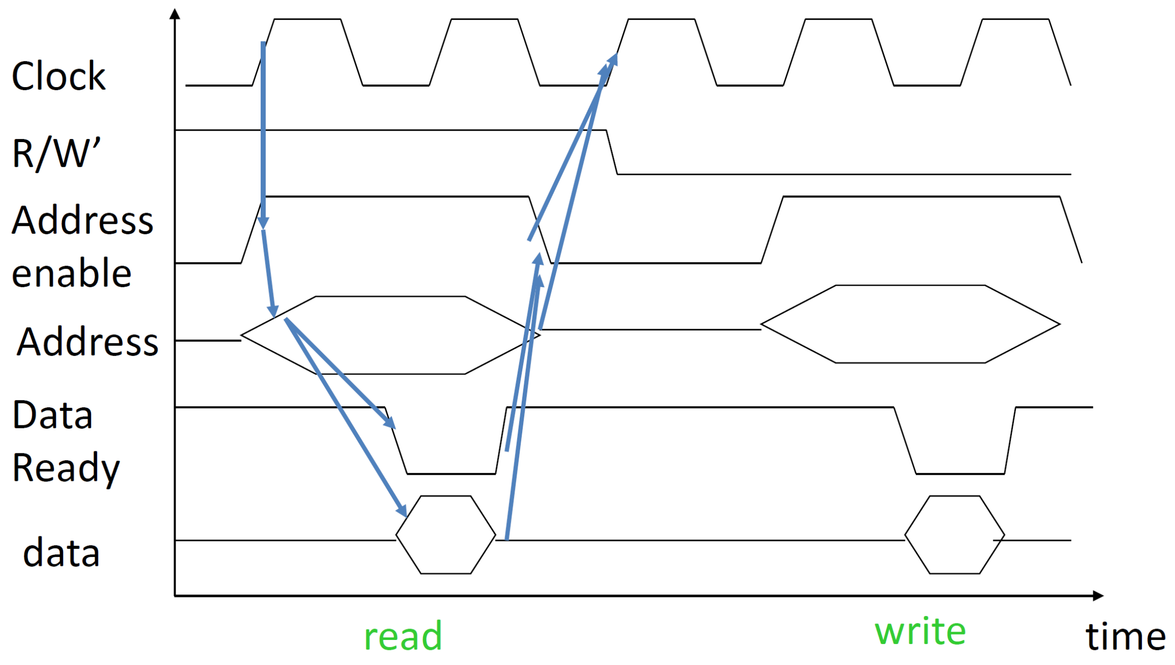
5.b

Cache should use SRAM for faster access. Main memory should use DRAM to achieve higher density.

5.c

Random access means data items can be read or written in almost the same amount of time irrespective of the physical location of data inside to memory. It is realized by multiplexing and demultiplexing circuitry to locate the data entries. Magnetic disk/tape doesn't support random access.

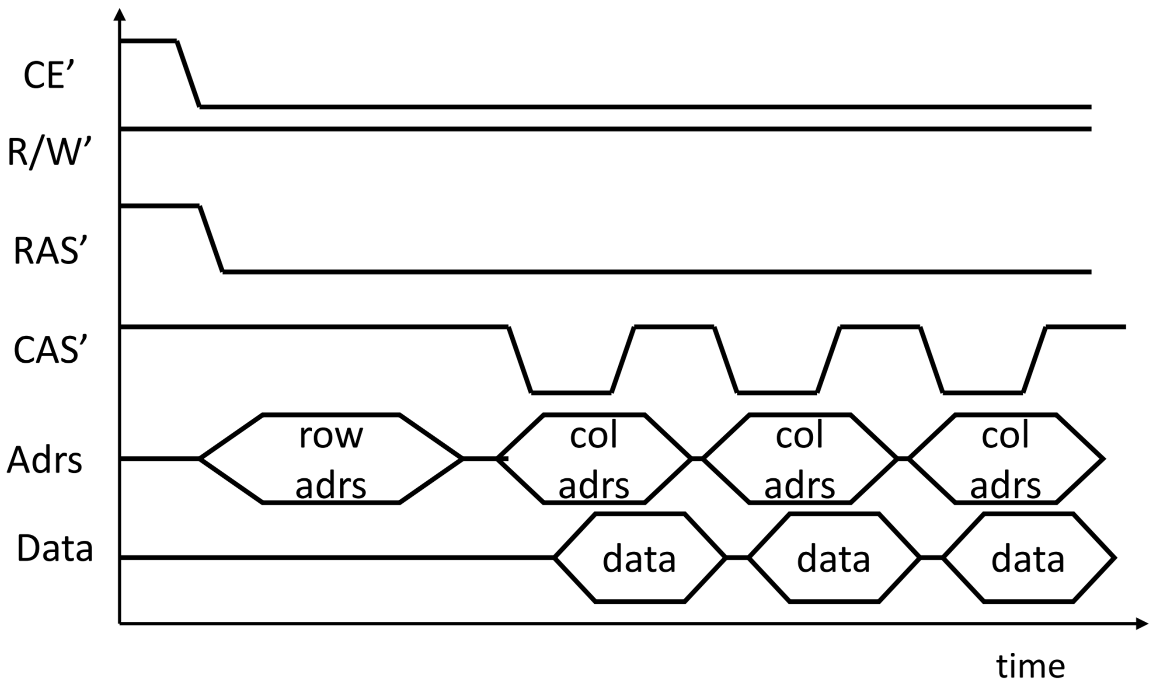
6.a



6.b

DRAM has to transmit row address and column address separately before data transfer. SRAM only sends one address before data transfer.

To improve DRAM speed, it should supply addressing with one row and many columns.



7.a

Parallel interfaces have less reliability due to mutual interference between wires. Serial I/O has almost zero mutual interference and can use higher clock frequency to increase transmission rate. For example, we use ethernet to realize high speed internet connection.

7.b

clock cycle duration

$$T_c = \frac{1}{1\ \mathrm{MHz}} = 10^{-6}\ \mathrm{s}$$

number of clock cycles to transmit 1kb data

$$N = (7+1+1) + (8+1) \times \frac{1000}{8} = 1134$$

Total time

$$t = NT_c = 1.134\ \mathrm{ms}$$

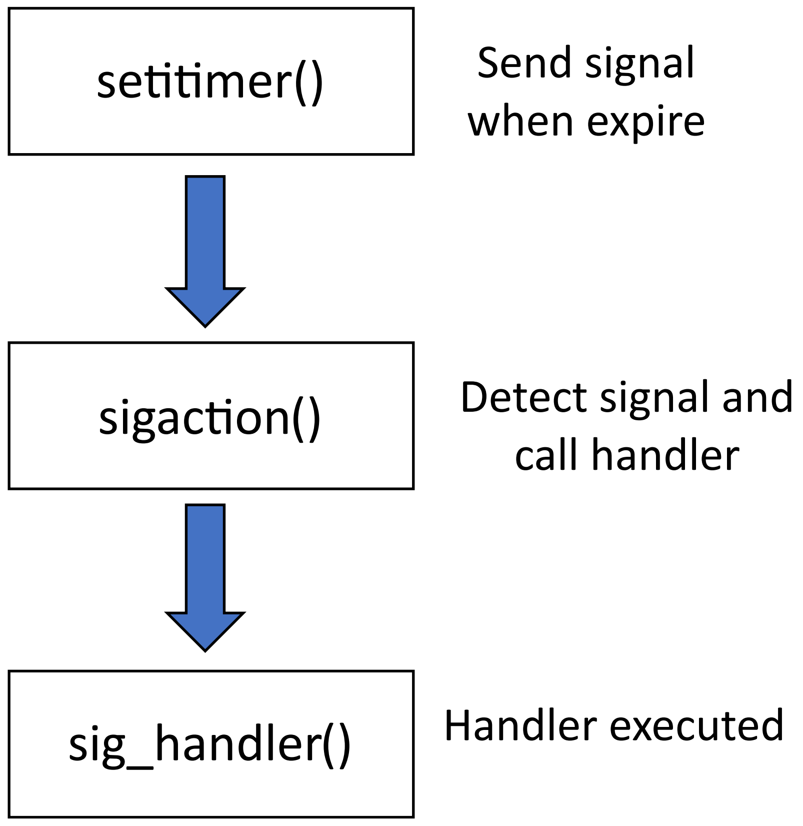
7.c

The key is SPI supports full duplex mode. And also SPI doesn't require ACK.

8.a

```
from sense_hat import SenseHat
import time
sense = SenseHat()
# 0.5 Hz
while True:
    for i in range(8):
        for j in range(8):
            setpixel(i,j,(0,0,255))
    sleep(1)
    sense.clear()
    sleep(1)
```

8.b



9.a

There will be a race condition.

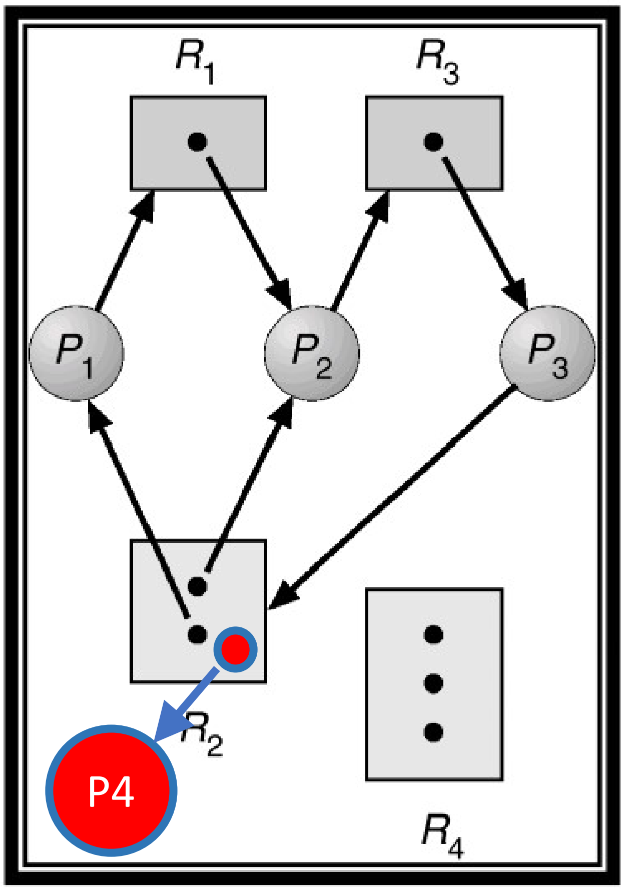
```
// define P and V operation
P(semaphore s){
    while(s == 0);
    s = s - 1;
}
V(semaphore s){
    s = s + 1;
}
// Initialize semaphore s as 1
s = 1;
// Each process does:
```

```
P(s);
if(flag == 0){
    flag = 1; loc = loc - 1;
}
V(s);
```

9.b

- R2 is held by P1 and P2.
- P1 is waiting for P2 to release R1.
- P2 is waiting for P3 to release R3.
- P3 is waiting for P1 and P2 to release R2.
- No extra resource point available in R1, R2 and R3 at this stage.

To remove the deadlock, we can simply add an extra ticket to either R1, R2, R3. Let's say we add to R2, such that once P4 releases it, P3 can immediately hold it and make the cycle alive.


R1
R3
P1
P2
P3
R2
P4
R4