Embedded Systems
Professor Thomas Bräunl
Associate Lecturer Kieran Quirke-Brown

Tutorial 4 – Architecture and ESP32

1. Archiutecture

Create a memory map of the following memory modules and then connect it back to the CPU. Note we are using 1 Byte words in memory. Use classic architecture.

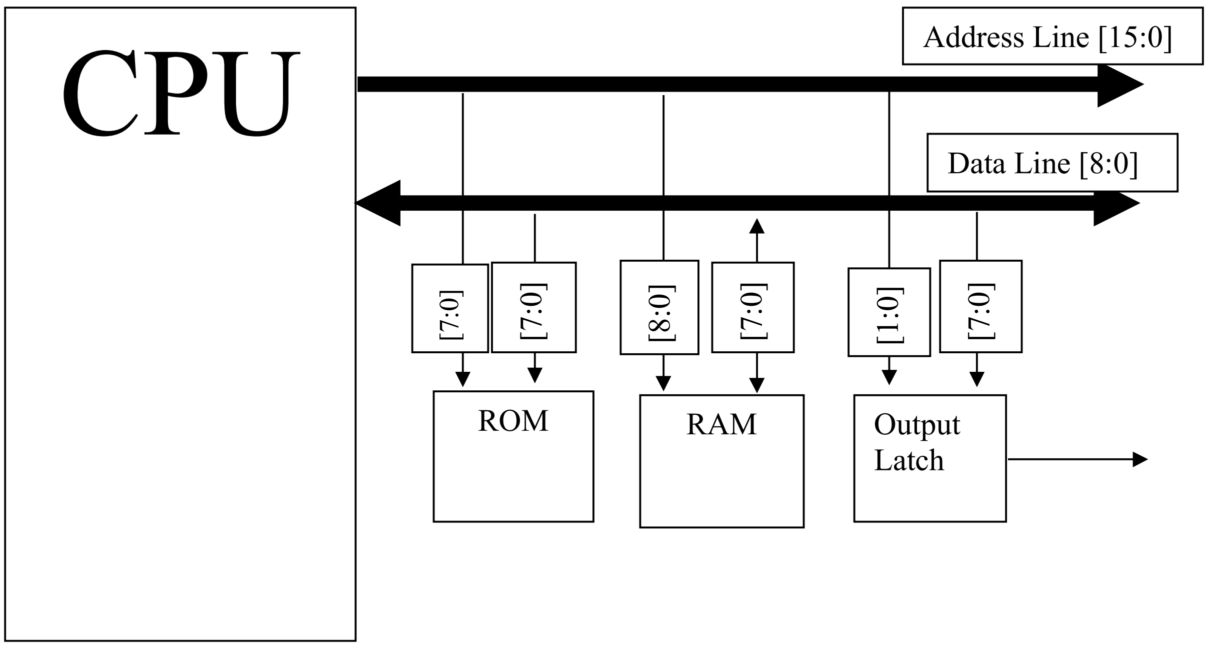
- ROM with 64 B
- RAM with 128 B
- Output Latch with 1 Byte

64 is 2 to the power of 6 which means we can represent all numbers from 000000 to 111111 or in hex 0x00 to 0x3F. Similarly for 128 we can represent all numbers from 0x00 to 0x7F and a latch will just be 1 address (i.e. 0x00 to 0x00). To create the memory block we need to make the memory continuous and we will start with the 64 bit memory (i.e. 0x00 – 0x3F). We then start the next memory block with 0x40, we can just offset the numbers from the 128B RAM by 0x40 to get 0x40 – 0xBF. We then start the latch from 0xC0 (again using the offset). So the memory map looks like:

ROM	0x00 – 0x3F
RAM	0x40 – 0xBF
Latch	0xC0 – 0xC0

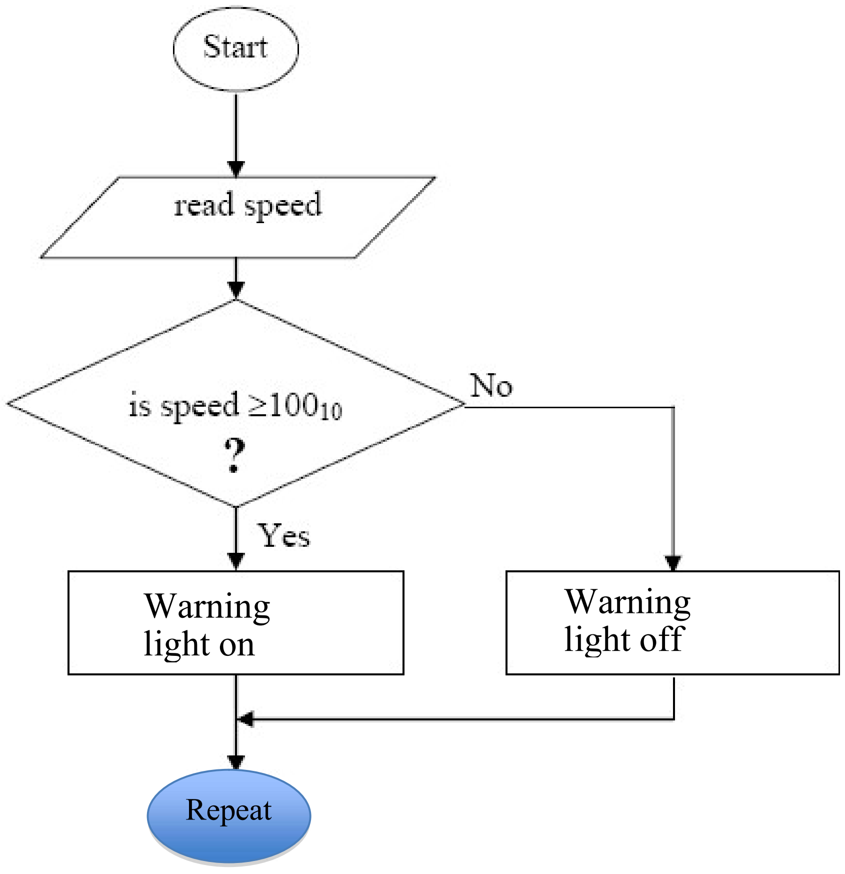
We then need to connect the memory to the CPU however there is an issue since the same address lines are attached we could have two chips trying to push different data out at the same time. We need to enable only one chip at a time. To do this we can look at the most significant bits. We know that when the latch is enabled that the top hex number will be 0xC and when written in binary that will be 1100. For the ROM the value can be between 0x0 and 0x3 so it could be 0000, 0001, 0011. For the RAM it can be between 0x4 and 0xB which can be written as 0100, 0101, 0110, 0111, 1000, 1001, 1010, 1011. We could use all 4 bits with combinatorial logic to act as our chip select lines but that would be inefficient. Lets see if we can find some common elements.

If we look at the two most significant bits we can see that the ROM is only on when both bits are 0. For the RAM it will be on if either of the bits are 1 but not both. Finally the Latch will be on whenever both bits are on. If we used combinatorial logic the ROM would use a NAND gate, the RAM would use an XOR and the Latch would use an and gate. Taking this into account then each memory module will need an extra two lines from the address line for the chip select. The ROM will need 8 lines (6 to address everything inside plus 2 for chip select), the RAM will need 9 and the latch will need only 2 (remember a latch only has an enable, data in and data out). This is shown below, remember to show the direction of data.



2. Data Monitoring

Write a **C subroutine** "checkspeed" for the ESP32 controller to perform the following function:



Speed is read as an analogue value from GPIO03.

If speed is greater or equal to 100 (decimal) it switches on a warning light; otherwise, the light is turned off. The warning light is controlled by GPIO02 otherwise the warning should be turned off and a fan should be ran at the corresponding speed (you do not need to send a signal for the fan). The speed encoders range is between 0 and 200 with a resolution of 0-1023 bits.

This program should loop forever.

Write a C program to solve this task.

There are many different solutions to this problem.

The input from the controller is a value between 0 – 1023 and the encoders range for reading the value is between 0 and 200. We need to convert between these values to make it work. Example solution below.

```
#define pinA 3
#define pinB 2

void setup() {
   pinMode(pinA, INPUT);
   pinMode(pinB, Output);
}

void loop() {
   int readValue = analogRead(pinA);
   float speed = (readValue/1023.0)*200.0;

   if (speed >= 100) {
      digitalWrite(pinB, HIGH);
   } else {
      digitalWrite(pinB, LOW);
   }
}
```

3. Temperature Control

A ESP32 microcontroller is used to control the temperature in this room. The current temperature of the room has been stored in flash memory address 0 and is stored as an integer in degrees Celsius. Also, a switch that controls the heater for this room is connected to GPIO01, and a switch that controls the air conditioner for this room is connected to GPIO02.

Using the airconditioner and heater the controller needs to maintain the temperature between 21 and 25 degrees. The system should use hystersis to limit the amount of switching. The system should also remember the current state of the airconditioner and heater in case power is lost.

Write a **C program** on the microcontroller.

This one is more complex as it is using flash memory, while it is good practice we wouldn't get you to write a program using flash memory. Again there are multiple different solutions for this. The best practice for programming is to just test your abilities with different scenarios, more can be found online.