EGC-121 SEC-A
COMPUTER ARCHITECTURE
PROJECT REPORT

BBC Micro Bit

Group Members:

1) Gourav Nayak – BT2024213
2) Sutaria Parth Anandkumar – BT2024028
3) Sricharan Anand – BT2024100
4) Madala Suhasini – BT2024043
5) Subhadeep Das – BT2024219
6) Sushmit Biswas – BT2024038
7) Thapan Komaravelly – BT2024076

OBJECTIVE & OVERVIEW

In this project, teams explored the fundamental concepts of Computer Architecture through the programming of a BBC micro:bit in ARM Assembly Language.

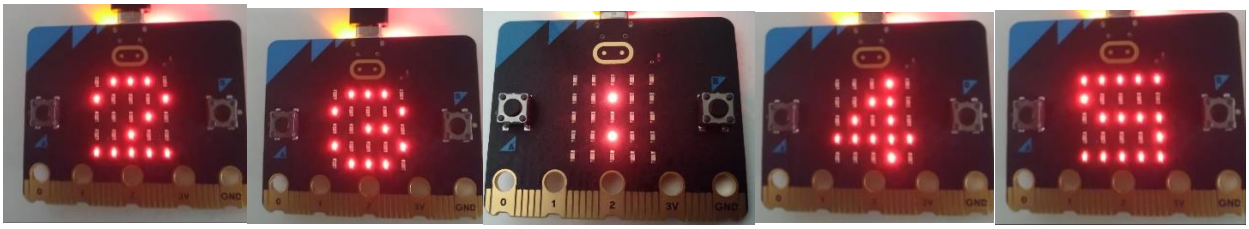
The assignment was open-ended.

Our team programmed an Alarm Clock on the BBC micro:bit completely in the ARM assembly language, using an audio library written by a former COMP2300 tutor, Benjamin Gray. The alarm clock is programmed to ring at a set time, along with a Snooze Function (to ring after 5 minutes) and a Dismiss Function.

The project allowed us to apply low-level computer architecture concepts through direct hardware control. Through this implementation, we gained hands-on experience with the following core computer architecture concepts:-

1) Memory-Mapped I/O: control of hardware peripherals (GPIO pins, LEDs, PWM) by reading/writing to memory addresses and registers
2) Polling: to check state of buttons
3) Interrupts: in the hardware timer
4) Stack Usage: in function calling

Please refer to this link for a video demo of our project.



The time 11:45 PM as displayed on the micro:bit. Each digit flashes for some time before moving to the next.

Between subsequent timestamps, there is a small gap where nothing is displayed on the LED matrix.

IMPLEMENTATION DETAILS

LED TIME DISPLAY

The time is displayed in the 24-hour format on the 5x5 LED Matrix, using binary patterns for all 10 digits (0-9), and a colon (:) to separate the hours and minutes. Each row is represented by a 32-bit .**word**, where each bit controls the LEDs' On/Off states in from left to right.

To display each digit, the program iterates through the rows of the LED matrix. The LEDs are controlled by its corresponding row and column pins. Since column pins are active low, a setting of 1 will turn off the LED, while 0 turns it on. Each digit is loaded into the registers, and the corresponding pins are activated to display the correct pattern.

The display_durations section of the code defines how long each digit and the colon are displayed. Each digit (representing the hours and minutes) is displayed for 38 units of time at a time. The colon (used as a separator between hours and minutes) is displayed for 50 units of time.

After displaying the entire time (hours and minutes), a delay of 63 units is introduced before the cycle repeats, during which the entire LED display is switched OFF. Each timestamp is shown 10 times per minute, after which the timestamp updates.

Register-to-immediate conditionals are implemented to handle digit carry-overs.

- The ones digits are reset to 0 after 9.
- The tens digit of the minutes is reset to 0 after 5.
- The hour digits are reset to 00 after 23 (23:59 is the maximum time).

The handling of pins with respect to the LEDs is covered by the external functions init_leds, write row_pins, write column_pins, and delay.

TIMING IMPLEMENTATION

During testing, it was determined experimentally that 43 units of time approximately equals 1 second real-time. This relationship is used throughout the implementation.

For example, the snooze duration of 5 minutes (300 seconds) is represented as 12900 units of time in this code.

AUDIO PLAYBACK

Digital sound is defined by samples, which move in a repeating saw-tooth pattern. Samples define how much a speaker cone should be pulled/pushed to make a sound in air.

The audio.s file enables sample-by-sample audio playback through Pulse Width Modulation (PWM). Rapid switching of a signal ON/OFF produces sound, with the duration of ON time (duty cycle) controlling the sound's frequency and volume.

The clock starts at a set timestamp. When the set time is arrived at, the alarm is played (showing a bell-pattern on the LED display) for a maximum of 60 seconds without any button presses. When Button A (pin 14) is pressed, the alarm is DISMISSED, and will never play till that timestamp is reached again. When Button B (pin 23) is pressed, the alarm is SNOOZED (showing a Z-pattern on the LED display), and will replay after 5 minutes. The correct time display is updated accordingly.

REGISTERS INVOLVED

- r0-r3: Parameter passing registers
    - r0: pin configurations, digit values
    - r1: hours part of time, alarm flags, durations
    - r2: minutes part of time, snooze duration, button states
    - r3: iteration counter
- r4-r7: alarm time (respective digits)
- r6: alarm/snooze flags, temp register in calculations, row counting
- r7: column patterns, alarm_settings
- r8-r11: current time (hours tens, hours ones, minutes tens, minutes ones digits)
- lr: return address in function calls
- sp: used inherently in pushes/pops.

CHALLENGES FACED

TIMING

Understanding the timing logic in ARM Assembly (wrt the BBC micro:bit) was tricky, as it wasn't based on conventional units.

To solve it, we experimented different times in code and used a real-time stopwatch to create a correlation. We found a relation, (units of time)/43 = seconds, that worked for the ranges of time required.

CLOCK UPDATES

After an alarm played for an entire minute without button intervention, the time display would not count the extra minute (if the alarm time was set at 23:46, it would only show 23:46, instead of 23:47). Similarly, for the snooze function, 5 minutes would not be counted. As a result, time would be lost without the user knowing it.

To solve it, we manually added the 1 minute and 5 minutes to the registers to be displayed. Since the code wasn't exactly configured to update the time in the background (of an alarm play or something), we found this to be the best possible solution.

SYMBOL DISPLAY

It was hard configuring the GPIO pins to the right registers. Even then the digits were distorted but after some trial and error it was showing up properly. Then we had to adjust how long each digit was displayed using separate variables for each digit and the colon and gap between consecutive timestamps.

FURTHER IMPROVEMENTS

- Setting the alarm time using the buttons.
- Allowing the user to set custom snooze intervals.
- Using other features like the touch logo, radio or Bluetooth to potentially communicate with other devices.
- Storing multiple alarm sounds.