BLG 212E - Microprocessor Systems

Homework 2 - Evaluation Criterias

Sadettin Fidan

January 10, 2025

Timer (50 points): `timing_asm.s`

- **SysTick_Handler (20 points)**:
  - Is the assembly implementation of `SysTick_Handler` in use and functioning correctly?
  - Score: $5 + 15 \times \frac{\text{Number of correctly commented lines}}{\text{Total number of lines}}$ points.
- **SysTick_Start_asm (15 points)**:
  - Is `SysTick_Start_asm` in use and functioning correctly?
  - Score: $5 + 10 \times \frac{\text{Number of correctly commented lines}}{\text{Total number of lines}}$ points.
- **SysTick_Stop_asm (15 points)**:
  - Is `SysTick_Stop_asm` in use and functioning correctly?
  - Score: $5 + 10 \times \frac{\text{Number of correctly commented lines}}{\text{Total number of lines}}$ points.

Sorting (40 points): `ft_lstsort_asm.s`

- **Sorting Functionality (5 points)**:
  - Does the implementation sort the data correctly?
- **Bubble Sort (15 points)**:
  - Does the implementation utilize the bubble sort algorithm and also use f_cmp which is given as argument? (5 points)

- Testing with two different f_cmp functions: ft_cmp2 and ft_cmp3. (5 x 2 = 10 points)

Listing 1: ft_cmp2, ft_cmp3

```c
int abs(int n)
{
        return (n >= 0 ? n :-n);
}

int ft_cmp2(int a, int b)
{
    int ah = a / 10;
    int al = a % 10;
    int bh = b / 10;
    int bl = b % 10;

    if (ah > bh)
        return 1;
    if (ah == bh && al < bl)
        return 1;
    return 0;
}
// n, n + 1, ..., n + 9,
// ...,
// 20, 21, 22, ..., 29,
// 10, 11, 12, ..., 19,
// 0, 1, 2, ..., 9.

int ft_cmp3(int a, int b)
{
        return abs(a) < abs(b);
}
// n+9,-(n+8), (n+8), ..., 5,-4, 3, 2,-1, 0.
```

- **Node Swapping (10 points)**:
  - Does the implementation correctly swap nodes?
- **Comment Quality (10 points)**:
  - Score: $10 \times \frac{\text{Number of correctly commented lines}}{\text{Total number of lines}}$ points.

BigO (10 points)

- **Data Points (2 points)**:
  - Are the data points {5, 10, 15, ..., 100}?
- **Graphs (2 points)**:
  - Does the figure include two graphs representing $O(n\log(n))$ and $O(n^2)$?
- **Time Measurements (6 points)**:
  - Do the time measurements align with the actual runtime of the code?

Deductions (8 points)

- **Built Files (2 points)**:
  - Are built files retained in the project directory?
- **Codes Folder (2 points)**:
  - Is there no folder named `codes` that contains the project source files?
- **Report File (2 points)**:
  - Is there no report file named `StudentNo_NameSurname_uP_HW2.pdf`?
- **File Organization (2 points)**:
  - Are the report file and the `codes` folder incorrectly located?

Final Grade Calculation

- Score = Timer + Sorting + BigO - Deductions
- Final Grade = max(0, Score)

Note

- Other edge cases in the evaluation:
  - If SysTick_Handler doesn't work properly (no ticks incerement, defining it both in C and ASM, etc.) then its score is 0, not 5 or more.
  - SysTick_Start_asm and SysTick_Stop_asm are actually evaluated together. So, any fault between these (e.g. no ticks measurement) two causes 0/30 points in Timing section.
  - If ft_lstsort_asm function causes any error (Hardfault, infinite loop, etc.), this section scores 0 despite solid logic and/or good quality commenting.
  - Empty/False graph with data points 5, 10, ..., 100 gives 0/10 in BigO section.
  - If the project is not compiled with ”rebuilt all target files” or compiles but doesn't go into debug mode, then Timing, Sorting, BigO parts all are 0 except:
    - SysTick_Handler is defined twice (e.g. both in ASM and in C), then C implementation is removed by us and then we try recompiling).
    - Some whitespace/indentation error. We omit this type of error. We fix the spacing issue, then try recompiling.
  - If naming of codes folder or report file is false, then it is also considered wrong file organization (Case sensivity is ommitted).
  - BigO sections is evaluated after both Timing and Sorting sections are ok.
  - The linked list is also an array as the area was preallocated and each node of linked list created and filled in in order. If one takes this advantage and uses the linked list as array and do sorting, Sorting section is 0/40, but BigO section will still be evaluated as if sorting works correctly.
  - If ft_lstsort_asm always sorts in reverse order of f_cmp behaviour, no point deduction just because of this situation.
  - If ft_cmp is KO, then ft_cmp2 is KO. Similarly, if ft_cmp2 is KO, ft_cmp3 also considered as KO (KO: Knock Out).
  - If n numbers given to ft_lstsort_asm, then sorted list should have exactly n elements, otherwise Sorting section is 0/40. If it is giving

n numbers with original myMain function but different number of elements with testing myMain function, still 0/40, but in this spesific edge case, BigO section still to be evaluated despite it was previously said that KO in any of Sorting or Timing sections makes BigO section KO as well.

Championship

- Who is Mr. Kunduracı?
- A sir who is the fastest bubble sorter.
- His ASM Bubble sort speed is so close to C merge sort.
- And the implementation:

Listing 2: ft_cmp2, ft_cmp3

```
; Function: ft_lstsort_asm
; Parameters:
;   R0 - Pointer to the list (address of t_list *)
;   R1 - Pointer to comparison function (address of int (*f_comp
            AREA    Sorting_Code, CODE, READONLY
    ALIGN
    THUMB
    EXPORT  ft_lstsort_asm

; Structure offsets for t_list
CONTENT_OFFSET  EQU     0       ; value offset
NEXT_OFFSET     EQU     4       ; next pointer offset

ft_lstsort_asm FUNCTION
    PUSH    {R4-R7, LR}

    MOV     R4, R0          ; R4 = list pointer
    MOV     R5, R1          ; R5 = comparison function

    ; Check empty or single node
    LDR     R1, [R4]        ; R1 = head pointer
    CMP     R1, #0
    BEQ     sort_done
    LDR     R2, [R1, #NEXT_OFFSET]
    CMP     R2, #0
    BEQ     sort_done

    outer_loop
        MOVS    R6, #0          ; R6 = swapped flag
        LDR     R1, [R4]        ; R1 = current node
        MOVS    R3, #0          ; R3 = previous node (NULL initially

    inner_loop
        LDR     R2, [R1, #NEXT_OFFSET]  ; R2 = next node
        CMP     R2, #0
        BEQ     check_swapped

        ; Compare values
        LDR     R0, [R1, #CONTENT_OFFSET]
        LDR     R7, [R2, #CONTENT_OFFSET]

        PUSH    {R1-R3, R4-R7}
        MOV     R1, R7
        BLX     R5              ; Call comparison function
        POP     {R1-R3, R4-R7}

        CMP     R0, #0
        BGT     no_swap

        ; Swap nodes
        LDR     R7, [R2, #NEXT_OFFSET]  ; R7 = next->next

        ; Update next pointers
        STR     R7, [R1, #NEXT_OFFSET]  ; current->next = next->next
        STR     R1, [R2, #NEXT_OFFSET]  ; next->next = current

        ; Update previous node s next pointer or head
        CMP     R3, #0
        BEQ     update_head
        STR     R2, [R3, #NEXT_OFFSET]  ; prev->next = next
        B       swap_done

    update_head
        STR     R2, [R4]        ; Update head pointer

    swap_done
        MOV     R1, R2          ; Move to next node
        MOVS    R6, #1          ; Set swapped flag
        B       continue_inner

    no_swap
        MOV     R3, R1          ; Save current as previous
        MOV     R1, R2          ; Move to next node

    continue_inner
        B       inner_loop

    check_swapped
        CMP     R6, #0
        BNE     outer_loop

    sort_done
        POP     {R4-R7, PC}
        ENDFUNC
        END
```