CSS422 Homework 4 Grade Rubric

| Q1 | 4pts | | | | Points | Your Grade |
| --- | --- | --- | --- | --- | --- | --- |
|  | 1-1:  **if ( R0 >= R1 ) { // 0.5pts**  **if ( ( R3 = R0 - R2 ) >= 0 ) { // 0.5pts, R3 assigned 0.5pts**  **R0 = 25;**  **}**  **}**  **else { // 0.5pts**  **if ( ( R4 = R2 - R0 ) < 0 ) // 0.5pts, R4 assigned 0.5pts**  **R0 = 15;**  **}** | | | | 3 | 2 |
| 1-2: 25 | | | | 0.5 | 0.5 |
| 1-3: 15 | | | | 0.5 | 0.5 |
| Q2 | 8pts | | | |  |  |
|  | **HW4\_2 LDR R0, =src1 ; R0 = 0x00000008**  **ADD R0, R0, #1 ; R0 = 0x00000009**  **LDR R1, [R0] ; R1 = 0xAAFF00FF**  **draw\_mem1 PUSH {R1} ; SP = 0x20000400->0x200003FC: (mem 0xAAFF00FF)**  **ADD R0, #4 ; R0 = 0x0000000D**  **LDRSH R2, [R0] ; R2 = 0xFFFFAA00, note this is a signed half word**  **draw\_mem2 PUSH {R2} ; SP = 0x200003FC->0x200003F8: (mem 0xFFFFAA00)**  **MOV R3, SP ; R3 = 0x200003F8**  **ADD R3, #3 ; R3 = 0x200003FB**  **LDR R4, [R3] ; R4 = 0xFF00FFFF**  **SUB R4, #0x004C004C; R4 = 0xFF00FFFF - 0x004C004C = 0xFEB4FFB3**  **draw\_mem3 STR R4, [R3] ; R3 = 0x200003FB (mem 0xFEB4FEB3)**  **POP {R2} ; SP = 0x20003F8->0x200003FC, R2 = 0xB3FFAA00**  **POP {R1} ; SP = 0x20003FC->0x20000400, R1 = 0xAAFEB4FF**  **Each mistake counts -0.25pts as you need to fill out 20 values.** | | | | 5 | 5 |
| **Address** | **draw mem1** | **draw mem2** | **Draw mem3** | 3 | 3 |
| **0x2000003F8** |  | **00** | **00** |
| **0x2000003F9** |  | **AA** | **AA** |
| **0x2000003FA** |  | **FF** | **FF** |
| **0x2000003FB** |  | **FF** | **B3** |
| **0x2000003FC** | **FF** | **FF** | **FE** |
| **0x2000003FD** | **00** | **00** | **B4** |
| **0x2000003FE** | **FF** | **FF** | **FE** |
| **0x2000003FF** | **AA** | **AA** | **AA** |
|  | **1pt**  **0.5pts: Wrong in 1-2 bytes** | **1pt**  **0.5pts: Wrong in 1-2 bytes** | **1pt**  **0.5pts: Wrong in 1-2 bytes** |  |  |
| Q3 | 8pts | | | |  |  |
|  | Source code  **AREA MyData, DATA, READWRITE**  **dst SPACE 200**  **AREA MyCODE, DATA, READONLY**  **src DCB "Computing and Software Systems, University of Washington Bothell", 0**  **AREA |.text|, CODE, READONLY**  **EXPORT \_\_main [WEAK]**  **ENTRY**  **\_\_main**  **LDR R0, =src ; src address**  **LDR R1, =dst ; dst address**  **; R2 = the current ASCCI Char**  **; R3 = counter from 7 to 0**  **; R4 = #ones**  **loop\_stmt**  **LDRB R2, [R0], #1 ; each src char\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 0.5pts**  **CBZ R2, end\_of\_main**  **MOV R3, #7 ; #bits (i.e., 7) in each ASCII**  **MOV R4, #0 ; clear #ones**  **loop\_char**  **RORS R2, #1 ; rotate left\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 0.5pts**  **ADDCS R4, #1 ; #ones++ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 1pt**  **SUBS R3, #1 ; counter-- \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 1pt**  **BNE loop\_char**    **ROR R2, R2, #25 ; retrieve the original ASCII char\_\_\_\_\_ 1pt**  **LSRS R4, #1 ; #ones % 2 == 1? (odd?)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 1pt**  **BCS odd\_number ; skip odd parity**  **ORR R2, #0x80 ; add a parity\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 1pt**  **odd\_number**  **STRB R2, [R1], #1 ; store the ASCII char with a parity\_\_\_\_ 0.5pts**  **B loop\_stmt**    **end\_of\_main**  **B end\_of\_main** | | | | 6.5 | 6.5 |
| Memory view (src) | | | | 0.5 | 0 |
| Memory view (dst) | | | | 1 | 0 |

Comments: Q3 – snapshots missing, total = 17.5