# LAB 5 REPORT

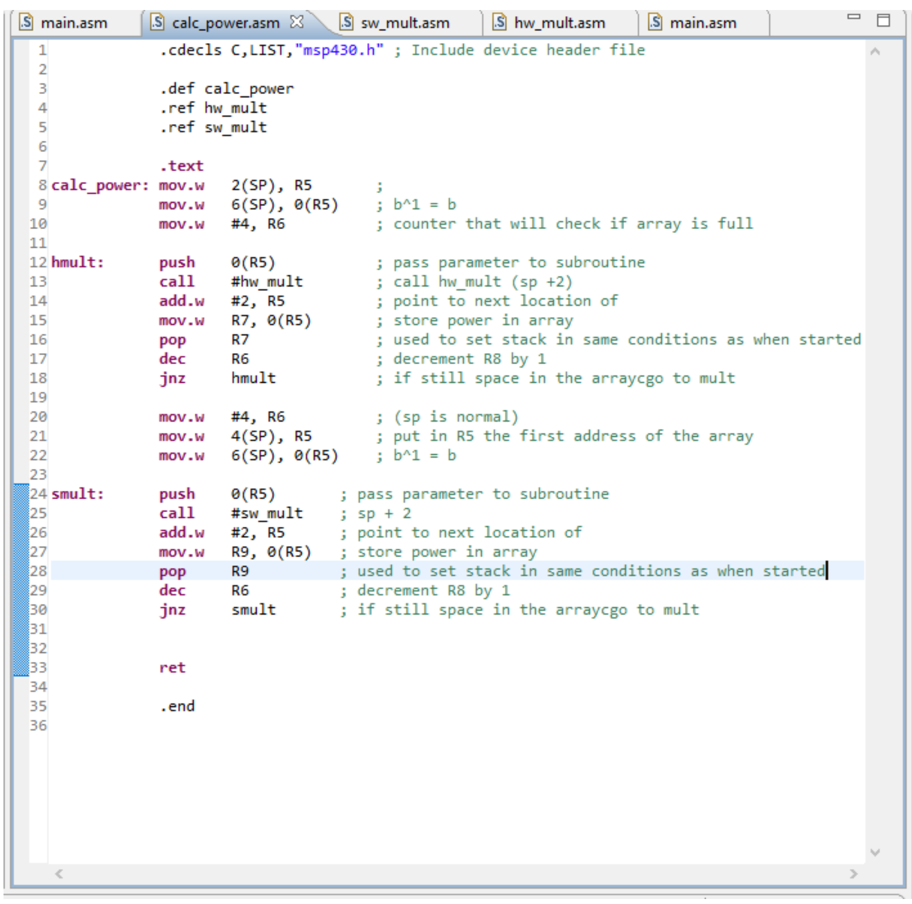
## 1. Theory topics:

a. **Passing parameters:** There are two ways to pass parameters. One is using general purpose registers. But this one has the problem that you may run out of registers and that is a problem.
The other way is using the stack. You place the data you want to use there with the instruction PUSH and you can access it whenever you want. The drawback of this method is that you have to keep track of the contents of the stack to know where your data is located. Also a thing to keep in min is that whenever you call a subroutine the stack pointer changes.

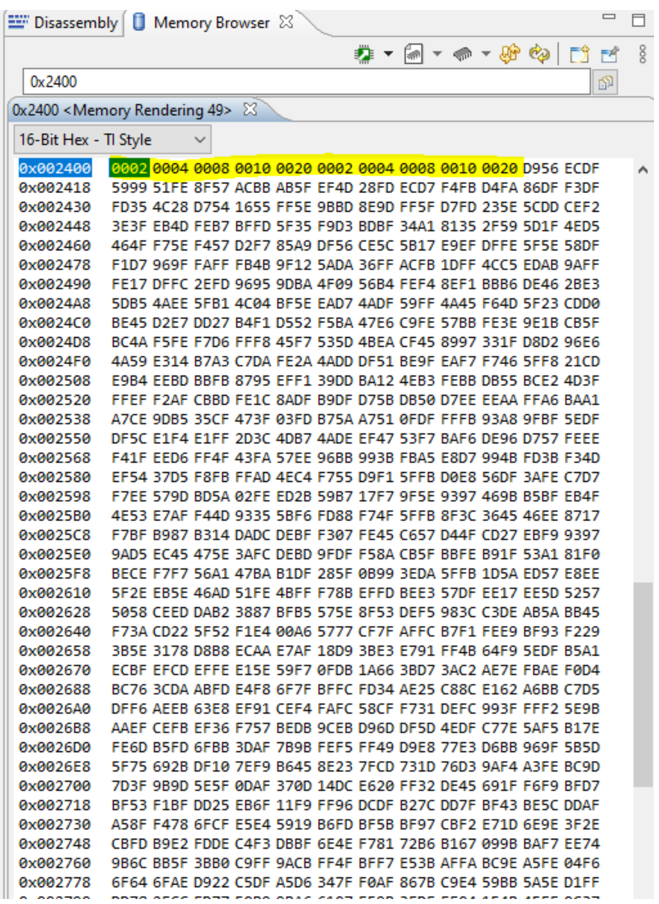
**b. Subroutines:** Subroutines are like functions when we use c or c++. When the program encounters the execution of a subroutines it stores the address of the program counter and executes the subprogram. Subroutines take in parameters and return another value. The way to access the subroutine is by using the word CALL and the name and to return no the main program or to the caller program we use ret.

**c. Hardware multiplier:** It is an optional peripheral that can perform multiplications using very few instructions. This is really handy since performing multiplications using logical operations is hard and expensive.

## 2. Program

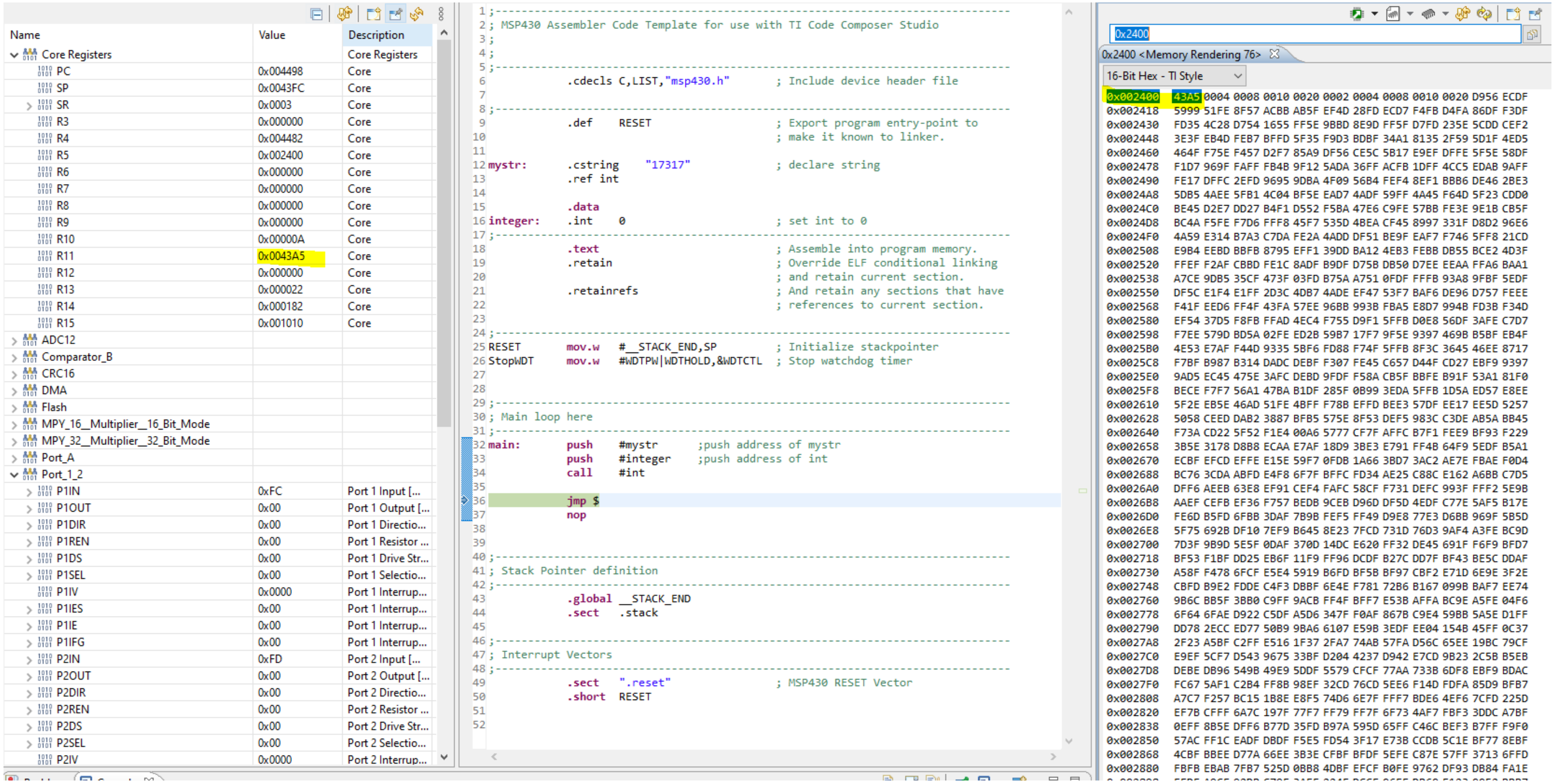


```
            .cdecls C,LIST,"msp430.h" ; Include device header file

            .def calc_power
            .ref hw_mult
            .ref sw_mult

            .text
calc_power: mov.w   2(SP), R5       ;
            mov.w   6(SP), 0(R5)    ; b^1 = b
            mov.w   #4, R6          ; counter that will check if array is full

hmult:      push    0(R5)           ; pass parameter to subroutine
            call    #hw_mult        ; call hw_mult (sp +2)
            add.w   #2, R5          ; point to next location of
            mov.w   R7, 0(R5)       ; store power in array
            pop     R7              ; used to set stack in same conditions as when started
            dec     R6              ; decrement R8 by 1
            jnz     hmult           ; if still space in the arraycgo to mult

            mov.w   #4, R6          ; (sp is normal)
            mov.w   4(SP), R5       ; put in R5 the first address of the array
            mov.w   6(SP), 0(R5)    ; b^1 = b

smult:      push    0(R5)       ; pass parameter to subroutine
            call    #sw_mult    ; sp + 2
            add.w   #2, R5      ; point to next location of
            mov.w   R9, 0(R5)   ; store power in array
            pop     R9          ; used to set stack in same conditions as when started
            dec     R6          ; decrement R8 by 1
            jnz     smult       ; if still space in the arraycgo to mult


            ret

            .end
```



```
0x002400   0002 0004 0008 0010 0020 0002 0004 0008 0010 0020 D956 ECDF
0x002418   5999 51FE 8F57 ACBB AB5F EF4D 28FD ECD7 F4FB D4FA 86DF F3DF
0x002430   FD35 4C28 D754 1655 FF5E 9BBD 8E9D FF5F D7FD 235E 5CDD CEF2
0x002448   3E3F EB4D FEB7 BFFD 5F35 F9D3 BDBF 34A1 8135 2F59 5D1F 4ED5
0x002460   464F F75E F457 D2F7 85A9 DF56 CE5C 5B17 E9EF DFFE 5F5E 58DF
0x002478   F1D7 969F FAFF FB4B 9F12 5ADA 36FF ACFB 1DFF 4CC5 EDAB 9AFF
0x002490   FE17 DFFC 2EFD 9695 9DBA 4F09 56B4 FEF4 8EF1 BBB6 DE46 2BE3
0x0024A8   5DB5 4AEE 5FB1 4C04 BF5E EAD7 4ADF 59FF 4A45 F64D 5F23 CDD0
0x0024C0   BE45 D2E7 DD27 B4F1 D552 F5BA 47E6 C9FE 57BB FE3E 9E1B CB5F
0x0024D8   BC4A F5FE F7D6 FFF8 45F7 535D 4BEA CF45 8997 331F D8D2 96E6
0x0024F0   4A59 E314 B7A3 C7DA FE2A 4ADD DF51 BE9F EAF7 F746 5FF8 21CD
0x002508   E9B4 EEBD BBFB 8795 EFF1 39DD BA12 4EB3 FEBB DB55 BCE2 4D3F
0x002520   FFEF F2AF CBBD FE1C 8ADF B9DF D75B DB50 D7EE EEAA FFA6 BAA1
0x002538   A7CE 9DB5 35CF 473F 03FD B75A A751 0FDF FFFB 93A8 9FBF 5EDF
0x002550   DF5C E1F4 E1FF 2D3C 4DB7 4ADE EF47 53F7 BAF6 DE96 D757 FEEE
0x002568   F41F EED6 FF4F 43FA 57EE 96BB 993B FBA5 E8D7 994B FD3B F34D
0x002580   EF54 37D5 F8FB FFAD 4EC4 F755 D9F1 5FFB D0E8 56DF 3AFE C7D7
0x002598   F7EE 579D BD5A 02FE ED2B 59B7 17F7 9F5E 9397 469B B5BF EB4F
0x0025B0   4E53 E7AF F44D 9335 5BF6 FD88 F74F 5FFB 8F3C 3645 46EE 8717
0x0025C8   F7BF B987 B314 DADC DEBF F307 FE45 C657 D44F CD27 EBF9 9397
0x0025E0   9AD5 EC45 475E 3AFC DEBD 9FDF F58A CB5F BBFE B91F 53A1 81F0
0x0025F8   BECE F7F7 56A1 47BA B1DF 285F 0B99 3EDA 5FFB 1D5A ED57 E8EE
0x002610   5F2E EB5E 46AD 51FE 4BFF F78B EFFD BEE3 57DF EE17 EE5D 5257
0x002628   5058 CEED DAB2 3887 BFB5 575E 8F53 DEF5 983C C3DE AB5A BB45
0x002640   F73A CD22 5F52 F1E4 00A6 5777 CF7F AFFC B7F1 FEE9 BF93 F229
0x002658   3B5E 3178 D8B8 ECAA E7AF 18D9 3BE3 E791 FF4B 64F9 5EDF B5A1
0x002670   ECBF EFCD EFFE E15E 59F7 0FDB 1A66 3BD7 3AC2 AE7E FBAE F0D4
0x002688   BC76 3CDA ABFD E4F8 6F7F BFFC FD34 AE25 C88C E162 A6BB C7D5
0x0026A0   DFF6 AEEB 63E8 EF91 CEF4 FAFC 58CF F731 DEFC 993F FFF2 5E9B
0x0026B8   AAEF CEFB EF36 F757 BEDB 9CEB D96D DF5D 4EDF C77E 5AF5 B17E
0x0026D0   FE6D B5FD 6FBB 3DAF 7B9B FEF5 FF49 D9E8 77E3 D6BB 969F 5B5D
0x0026E8   5F75 692B DF10 7EF9 B645 8E23 7FCD 731D 76D3 9AF4 A3FE BC9D
0x002700   7D3F 9B9D 5E5F 0DAF 370D 14DC E620 FF32 DE45 691F F6F9 BFD7
0x002718   BF53 F1BF DD25 EB6F 11F9 FF96 DCDF B27C DD7F BF43 BE5C DDAF
0x002730   A58F F478 6FCF E5E4 5919 B6FD BF5B BF97 CBF2 E71D 6E9E 3F2E
0x002748   CBFD B9E2 FDDE C4F3 DBBF 6E4E F781 72B6 B167 099B BAF7 EE74
0x002760   9B6C BB5F 3BB0 C9FF 9ACB FF4F BFF7 E53B AFFA BC9E A5FE 04F6
0x002778   6F64 6FAE D922 C5DF A5D6 347F F0AF 867B C9E4 59BB 5A5E D1FF
```

**Note: input is 2.**



| Name | Value | Description |
|---|---|---|
| Core Registers | | Core Registers |
| PC | 0x004498 | Core |
| SP | 0x0043FC | Core |
| SR | 0x0003 | Core |
| R3 | 0x000000 | Core |
| R4 | 0x004482 | Core |
| R5 | 0x002400 | Core |
| R6 | 0x000000 | Core |
| R7 | 0x000000 | Core |
| R8 | 0x000000 | Core |
| R9 | 0x000000 | Core |
| R10 | 0x00000A | Core |
| R11 | 0x0043A5 | Core |
| R12 | 0x000000 | Core |
| R13 | 0x000022 | Core |
| R14 | 0x000182 | Core |
| R15 | 0x001010 | Core |
| ADC12 | | |
| Comparator_B | | |
| CRC16 | | |
| DMA | | |
| Flash | | |
| MPY_16__Multiplier__16_Bit_Mode | | |
| MPY_32__Multiplier__32_Bit_Mode | | |
| Port_A | | |
| Port_1_2 | | |
| P1IN | 0xFC | Port 1 Input [... |
| P1OUT | 0x00 | Port 1 Output [... |
| P1DIR | 0x00 | Port 1 Directio... |
| P1REN | 0x00 | Port 1 Resistor ... |
| P1DS | 0x00 | Port 1 Drive Str... |
| P1SEL | 0x00 | Port 1 Selectio... |
| P1IV | 0x0000 | Port 1 Interrup... |
| P1IES | 0x00 | Port 1 Interrup... |
| P1IE | 0x00 | Port 1 Interrup... |
| P1IFG | 0x00 | Port 1 Interrup... |
| P2IN | 0xFD | Port 2 Input [... |
| P2OUT | 0x00 | Port 2 Output [... |
| P2DIR | 0x00 | Port 2 Directio... |
| P2REN | 0x00 | Port 2 Resistor ... |
| P2DS | 0x00 | Port 2 Drive Str... |
| P2SEL | 0x00 | Port 2 Selectio... |
| P2IV | 0x0000 | Port 2 Interrup... |

```
;-------------------------------------------------------------------------------
; MSP430 Assembler Code Template for use with TI Code Composer Studio
;
;
;-------------------------------------------------------------------------------
            .cdecls C,LIST,"msp430.h"       ; Include device header file

;-------------------------------------------------------------------------------
            .def    RESET                   ; Export program entry-point to
                                            ; make it known to linker.

mystr:      .cstring    "17317"             ; declare string
            .ref int

            .data
integer:    .int    0                       ; set int to 0
;-------------------------------------------------------------------------------
            .text                           ; Assemble into program memory.
            .retain                         ; Override ELF conditional linking
                                            ; and retain current section.
            .retainrefs                     ; And retain any sections that have
                                            ; references to current section.

;-------------------------------------------------------------------------------
RESET       mov.w   #__STACK_END,SP         ; Initialize stackpointer
StopWDT     mov.w   #WDTPW|WDTHOLD,&WDTCTL  ; Stop watchdog timer


;-------------------------------------------------------------------------------
; Main loop here
;-------------------------------------------------------------------------------
main:       push    #mystr      ;push address of mystr
            push    #integer    ;push address of int
            call    #int

            jmp $
            nop


;-------------------------------------------------------------------------------
; Stack Pointer definition
;-------------------------------------------------------------------------------
            .global __STACK_END
            .sect   .stack

;-------------------------------------------------------------------------------
; Interrupt Vectors
;-------------------------------------------------------------------------------
            .sect   ".reset"                ; MSP430 RESET Vector
            .short  RESET

```

0x2400

0x2400 <Memory Rendering 76>

16-Bit Hex - TI Style

```
0x002400   43A5 0004 0008 0010 0020 0002 0004 0008 0010 0020 D956 ECDF
0x002418   5999 51FE 8F57 ACBB AB5F EF4D 28FD ECD7 F4FB D4FA 86DF F3DF
0x002430   FD35 4C28 D754 1655 FF5E 9BBD 8E9D FF5F D7FD 235E 5CDD CEF2
0x002448   3E3F EB4D FEB7 BFFD 5F35 F9D3 BDBF 34A1 8135 2F59 5D1F 4ED5
0x002460   464F F75E F457 D2F7 85A9 DF56 CE5C 5B17 E9EF DFFE 5F5E 58DF
0x002478   F1D7 969F FAFF FB4B 9F12 5ADA 36FF ACFB 1DFF 4CC5 EDAB 9AFF
0x002490   FE17 DFFC 2EFD 9695 9DBA 4F09 56B4 FEF4 8EF1 BBB6 DE46 2BE3
0x0024A8   5DB5 4AEE 5FB1 4C04 BF5E EAD7 4ADF 59FF 4A45 F64D 5F23 CDD0
0x0024C0   BE45 D2E7 DD27 B4F1 D552 F5BA 47E6 C9FE 57BB FE3E 9E1B CB5F
0x0024D8   BC4A F5FE F7D6 FFF8 45F7 535D 4BEA CF45 8997 331F D8D2 96E6
0x0024F0   4A59 E314 B7A3 C7DA FE2A 4ADD DF51 BE9F EAF7 F746 5FF8 21CD
0x002508   E9B4 EEBD BBFB 8795 EFF1 39DD BA12 4EB3 FEBB DB55 BCE2 4D3F
0x002520   FFEF F2AF CBBD FE1C 8ADF B9DF D75B DB50 D7EE EEAA FFA6 BAA1
0x002538   A7CE 9DB5 35CF 473F 03FD B75A A751 0FDF FFFB 93A8 9FBF 5EDF
0x002550   DF5C E1F4 E1FF 2D3C 4DB7 4ADE EF47 53F7 BAF6 DE96 D757 FEEE
0x002568   F41F EED6 FF4F 43FA 57EE 96BB 993B FBA5 E8D7 994B FD3B F34D
0x002580   EF54 37D5 F8FB FFAD 4EC4 F755 D9F1 5FFB D0E8 56DF 3AFE C7D7
0x002598   F7EE 579D BD5A 02FE ED2B 59B7 17F7 9F5E 9397 469B B5BF EB4F
0x0025B0   4E53 E7AF F44D 9335 5BF6 FD88 F74F 5FFB 8F3C 3645 46EE 8717
0x0025C8   F7BF B987 B314 DADC DEBF F307 FE45 C657 D44F CD27 EBF9 9397
0x0025E0   9AD5 EC45 475E 3AFC DEBD 9FDF F58A CB5F BBFE B91F 53A1 81F0
0x0025F8   BECE F7F7 56A1 47BA B1DF 285F 0B99 3EDA 5FFB 1D5A ED57 E8EE
0x002610   5F2E EB5E 46AD 51FE 4BFF F78B EFFD BEE3 57DF EE17 EE5D 5257
0x002628   5058 CEED DAB2 3887 BFB5 575E 8F53 DEF5 983C C3DE AB5A BB45
0x002640   F73A CD22 5F52 F1E4 00A6 5777 CF7F AFFC B7F1 FEE9 BF93 F229
0x002658   3B5E 3178 D8B8 ECAA E7AF 18D9 3BE3 E791 FF4B 64F9 5EDF B5A1
0x002670   ECBF EFCD EFFE E15E 59F7 0FDB 1A66 3BD7 3AC2 AE7E FBAE F0D4
0x002688   BC76 3CDA ABFD E4F8 6F7F BFFC FD34 AE25 C88C E162 A6BB C7D5
0x0026A0   DFF6 AEEB 63E8 EF91 CEF4 FAFC 58CF F731 DEFC 993F FFF2 5E9B
0x0026B8   AAEF CEFB EF36 F757 BEDB 9CEB D96D DF5D 4EDF C77E 5AF5 B17E
0x0026D0   FE6D B5FD 6FBB 3DAF 7B9B FEF5 FF49 D9E8 77E3 D6BB 969F 5B5D
0x0026E8   5F75 692B DF10 7EF9 B645 8E23 7FCD 731D 76D3 9AF4 A3FE BC9D
0x002700   7D3F 9B9D 5E5F 0DAF 370D 14DC E620 FF32 DE45 691F F6F9 BFD7
0x002718   BF53 F1BF DD25 EB6F 11F9 FF96 DCDF B27C DD7F BF43 BE5C DDAF
0x002730   A58F F478 6FCF E5E4 5919 B6FD BF5B BF97 CBF2 E71D 6E9E 3F2E
0x002748   CBFD B9E2 FDDE C4F3 DBBF 6E4E F781 72B6 B167 099B BAF7 EE74
0x002760   9B6C BB5F 3BB0 C9FF 9ACB FF4F BFF7 E53B AFFA BC9E A5FE 04F6
0x002778   6F64 6FAE D922 C5DF A5D6 347F F0AF 867B C9E4 59BB 5A5E D1FF
0x002790   DD78 2ECC ED77 50B9 9BA6 6107 E59B 3EDF EE04 154B 45FF 0C37
0x0027A8   2F23 A5BF C2FF E516 1F37 2FA7 74AB 57FA D56C 65EE 19BC 79CF
0x0027C0   E9EF 5CF7 D543 9675 33BF D204 4237 D942 E7CD 9B23 2C5B B5EB
0x0027D8   DEBE DB96 549B 49E9 5DDF 5579 CFCF 77AA 733B 6DF8 EBF9 BDAC
0x0027F0   FC67 5AF1 C2B4 FF8B 98EF 32CD 76CD 5EE6 F14D FDFA 85D9 BFB7
0x002808   A7C7 F257 BC15 1B8E E8F5 74D6 6E7F FFF7 BDE6 4EF6 7CFD 225D
0x002820   EF7B CFFF 6A7C 197F 77F7 FF79 FF7F 6F73 4AF7 FBF3 3DDC A7BF
0x002838   0EFF 8B5E DFF6 B77D 35FD B97A 595D 65FF C46C BEF3 B7FF F9F0
0x002850   57AC FF1C EADF DBDF F5E5 FD54 3F17 E73B CCDB 5C1E BF77 8EBF
0x002868   4CBF BBEE D77A 66EE 3B3E CFBF BFDF 5EFE C87E 57FF 3713 6FFD
0x002880   FBFB EBAB 7FB7 525D 0BB8 4DBF EFCF B0FE 9762 DF93 DB84 FA1E
```