;r0 output

;r4 input to table

;r5 input

;r7 input

;r1 + 2 input

;r10 used for address of the subroutine call

                                .data.b h'64                              ;00 hex 64

                                .data.b h'64                              ;01 hex 64

                                .data.b h'63                              ;02 hex 63

                                .data.b h'63                              ;03 hex 63

                                .data.b h'62                              ;04 hex 62

                                .data.b h'61                              ;05 hex 61

                                .data.b h'60                              ;06 hex 60

                                .data.b h'58                              ;07 hex 58

                                .data.b h'44                              ;08 hex 44

                                .data.b h'42                              ;09 hex 42

                                .data.b h'42                              ;0a hex 42

                                .data.b h'42                              ;0b hex 42

                                .data.b h'42                              ;0c hex 42

                                .data.b h'42                              ;0d hex 42

                                .data.b h'42                              ;0e hex 42

                                .data.b h'41                              ;0f hex 41

                                .data.b h'40                              ;10 hex 40

                                .data.b h'40                              ;11 hex 40

                                .data.b h'40                              ;12 hex 40

                                .data.b h'40                              ;13 hex 40

                                .data.b h'3F                              ;14 hex 3f

; FUNC\_UNK1: function compares a byte to 0xf0

; r5 has an unknown value, r0 has a value of 0xf0 (240 dec)

; if r5 >= r0 is TRUE

;               ^-it will branch to loc\_2978A and return a value of 0x41 from the data structure

; continue

; make a copy of r5 into r3

; r5 is shifted right 4 bits to become the bottom nibble even though it was originally the (MSBs)

; r0 is then inverted into r1 which turns 0xf0 into 0x0f

; r1 becomes a mask then logical AND is performed on r3, results in only values in the lower nibble

; r3 is shifted left 12 bits then becomes the MSB even though originally it was the (LSB)

; r4 currently points to the address of the data structure, r5 has some offset that it wants to read. store the new value, r4 = (r5 + r4)

; at this point r4 and r5 currently have unknown values but we know r0 = 0xf0, r1 = 0x0f, r3 = (shifted value of r5)

; next, the data from table is going to be retrieved from (r4 + offset 0) and stored into r1

; next, the data from table is going to be retrieved from (r4 + offset 1) and stored into r0

; r1 and r0 is then zero byte extended to reduce both registers to a byte

; if r0 >= r1 is TRUE

;               ^-it will branch to loc\_29677

;                 r0 - r1, r0 will net a positive result

;                 then, r0 x r3, result is stored in MACL

;                 retrieve the 32 bit value from MACL and store it into r0

;                 shift r0 right 16 bits

;                 add r1 to r0 and return r0

; continue

; backup r1 (the value larger than r0) into r7

; r1 - r0, r1 will net a positive result

; then, r1 x r3, result is stored in MACL

; retrieve the value from MACL and store into r1

; restore r7 into r0 (which was the original value in r1 a few lines up)

; shift r1 right 16 bits

; subtract r0 - r1, return r0