1. Given code in assembly

```
.global _start
_start: br Start  # begin at the main program

# Constants
  .equ HEX_DISP, 0x88A0
  .equ HEX_CONT, 0x88B0

.org 0x0100

Start:
# r2 is the counter
  init:
    # enable all hex displays
    movia r3, HEX_CONT
    movi r4, 0xFF  # enable all hex displays
    stw r4, 0(r3)  # store to HEX_CONT

    movia r5, HEX_DISP  # r5 points to the first hex display
    movi r2, 0  # initialize counter to 0
  loop:
    stw r2, 0(r5)  # store the counter value to the hex display
    addi r2, r2, 1  # increment the counter
    br loop  # repeat the loop
```

2. Subroutines

```c
void outchar(char ch){
  volatile int *const UART = (int *)0x00008840; // UART register
  *UART = ch; // send character to UART
}

char bin2hex(char N){
// take 4 LSB of N, convert to hex, return
  N &= 0x0F; // mask to get 4 LSB
  if (N < 10) {

    printf("bin2hex: Output = %c\n", N + 0x30); // Debug print

    return N + '0'; // convert to ASCII for digits 0-9
  }
  else {

    printf("bin2hex: Output = %c\n", N + 0x37); // Debug print

    return N - 10 + 'A'; // convert to ASCII for letters A-F
  }
}

void outhex(char N){
  outchar(bin2hex((N & 0xF0) >> 4)); // send first hex digit
  outchar(bin2hex(N & 0x0F)); // send first hex digit
}

```

The meaning of void in front of outhex(char N) means it doesn’t return a value from the subroutine.