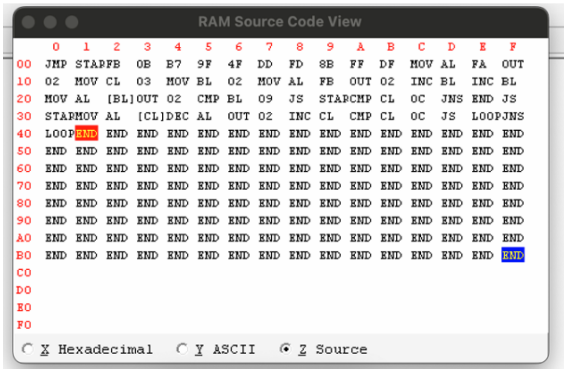
Name: SOUNG VICHKA

ID: e20240672

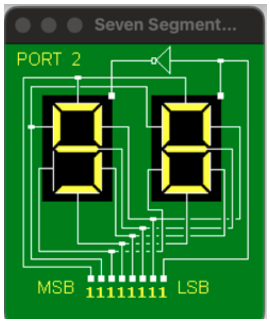
Group: I2-GIC2-B

TP10

a) Displays even numbers: Ex: 00, 02, 04, …., 98

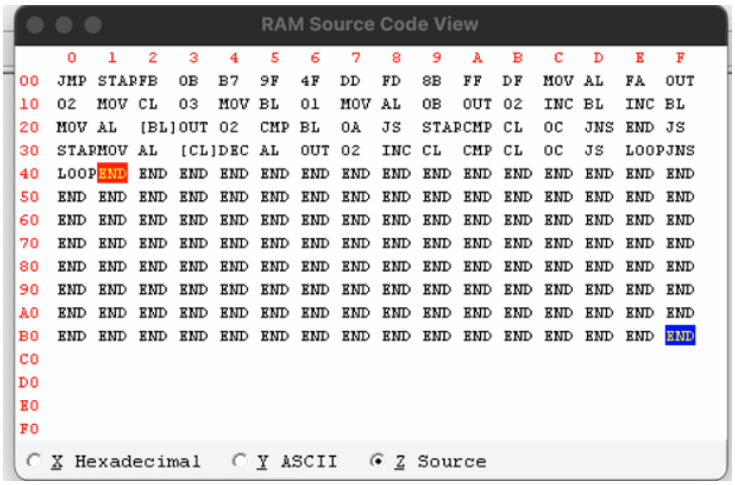



```
JMP start          ; Jump over data definitions to start label
; Seven-segment display encodings for digits 0-9:
DB FB ;0
DB 0B ;1
DB B7 ;2
DB 9F ;3
DB 4F ;4
DB DD ;5
DB FD ;6
DB BB ;7
DB FF ;8
DB DF ;9
start:
MOV AL,FA          ; Initialize display control (AL = 0xFA)
OUT 02             ; Send AL to port 2 (display control)
MOV CL,03          ; Initialize counter (CL = 3) [points to digit 3 in memory]
LOOP:
MOV BL,02          ; Initialize digit pointer (BL = 2) [points to digit 2 in memory]
start2:
MOV AL, FB         ; Load digit separator (AL = 0xFB)
OUT 02             ; Output to port 2
start3:
INC BL             ; Increment digit pointer (BL = BL + 1)
INC BL             ; Increment digit pointer again since we want an even number (BL = BL + 2 total)
MOV AL,[BL]        ; Load digit pattern from memory (AL = memory[BL])
OUT 02             ; Output digit pattern to port 2
CMP BL, 09         ; If BL < 9 (signed), jump to start3
```

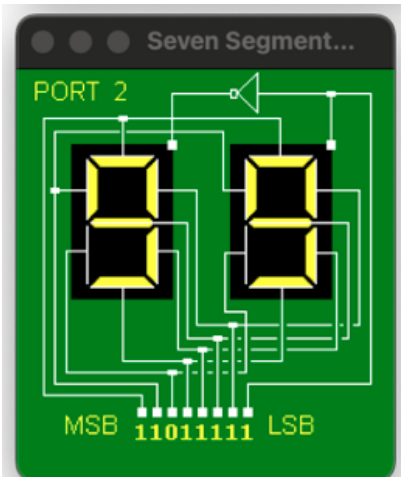


```
JS start3
CMP CL, 0C      ; Compare CL with 12 (0x0C)
JNS end         ; If CL >= 12 (not signed), jump to end
JS start4       ; If CL < 12 (signed), jump to start4
start4:
MOV AL,[CL]     ; Load current tens digit (AL = memory[CL])
DEC AL          ; Decrement to get correct digit (AL = AL - 1)
OUT 02          ; Output to port 2
INC CL          ; Increment counter (CL = CL + 1)
CMP CL, 0C      ; Compare CL with 12
JS LOOP         ; If CL < 12 (signed), jump to LOOP
JNS LOOP        ; If CL >= 12 (not signed), jump to LOOP [Redundant]
end:
END
```

b) Displays odd numbers: Ex: 01, 03, 05,…..,99



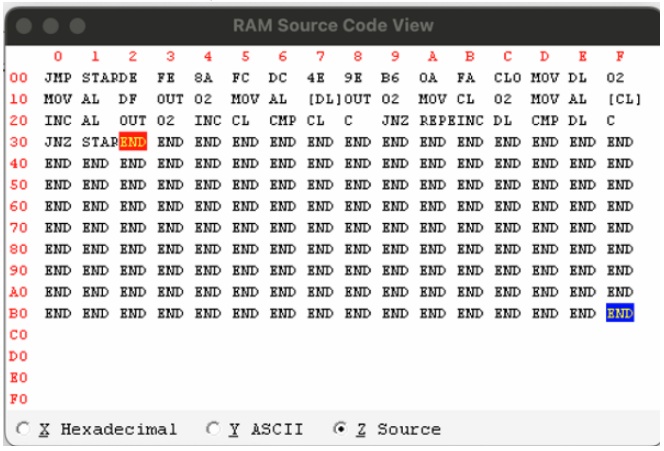

```
JMP start       ; Jump over data definitions to start of program
; Seven-segment display encodings for digits 0-9:
DB FB ; 0
DB 0B ; 1
DB B7 ; 2
DB 9F ; 3
DB 4F ; 4
DB DD ; 5
DB FD ; 6
DB BB ; 7
DB FF ; 8
DB DF ; 9
start:
MOV AL, FA      ; Load display control value (AL = 0xFA)
OUT 02          ; Output to port 2 (display control)
MOV CL, 03      ; Initialize counter to 3 (points to digit 3 in memory)
LOOP:
MOV BL, 01      ; Initialize digit pointer to 1 (points to digit 1 in memory)
start2:
MOV AL, 0B      ; Load digit separator pattern (AL = 0x0B)
OUT 02          ; Output separator to port 2
start3:
```



```
INC BL          ; Increment digit pointer (BL = BL + 1)
INC BL          ; Increment digit pointer again (BL = BL + 1)
MOV AL, [BL]    ; Load digit pattern from memory at address BL
OUT 02          ; Output digit pattern to port 2
CMP BL, 0A      ; Compare BL with 10 (0x0A)
JS start3       ; If BL < 10 (not signed), jump to start3
CMP CL, 0C      ; Compare CL with 12 (0x0C)
JNS end         ; If CL >= 12 (not signed), jump to end
JS start4       ; If CL < 12 (signed), jump to start4
start4:
MOV AL, [CL]    ; Load current tens digit pattern from memory at address CL
DEC AL          ; Decrement to adjust digit value (AL = AL - 1)
OUT 02          ; Output adjusted digit to port 2
MOV AL, FB      ; Load separator pattern (0xFB) into AL
OUT 02          ; Output separator to display
INC CL          ; Increment counter (CL = CL + 1)
CMP CL, 0C      ; Compare CL with 12 again
JS LOOP         ; If CL < 12 (signed), jump to LOOP
JNS LOOP        ; If CL >= 12 (not signed), jump to LOOP [Note: This is redundant]
end:
END
```

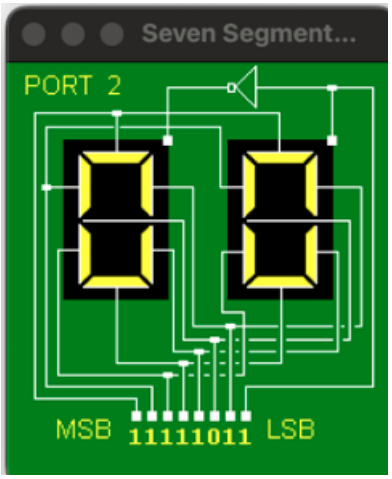
c) Displays countdown: Ex: 99, 98, 97, ….., 00

```
JMP STARTO      ; Jump over data definitions to START label
; Seven-segment display encodings for digits (values may need verification):
DB DE ; 9
DB FE ; 8
DB 9A ; 7
DB FC ; 6
DB DC ; 5
DB 4E ; 4
DB 9E ; 3
DB B6 ; 2
DB 0A ; 1
DB FA ; 0
```




```
STARTO:
CLO             ; Close all peripherals (clean start)
MOV DL, 02      ; Initialize DL to 02 (pointer to second digit pattern)
```

START:
MOV AL, DF ; Load AL with display control value (0xDF)
OUT 02 ; Send control value to port 2 (display control)
MOV AL, [DL] ; Load digit pattern from memory at address DL
OUT 02 ; Output digit pattern to port 2



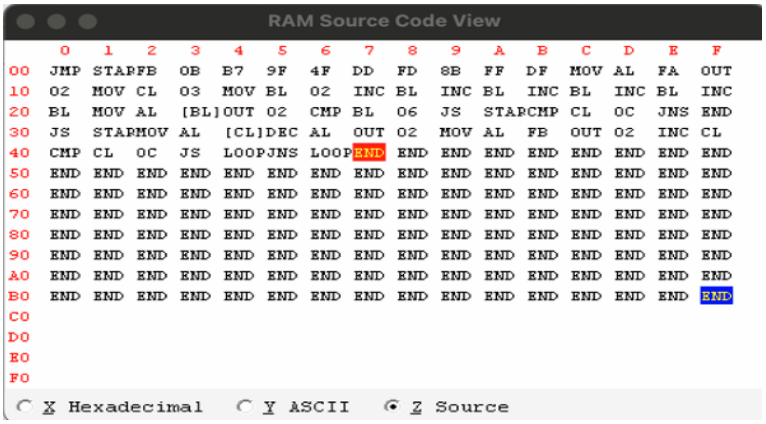
MOV CL, 02 ; Initialize counter CL to 02 (starting position)
REPEAT:
MOV AL, [CL] ; Load digit pattern from memory at address CL
INC AL ; Increment the digit pattern (modify display)
OUT 02 ; Output modified pattern to port 2
INC CL ; Increment memory pointer (CL = CL + 1)
CMP CL, C ; Compare CL with 0xC (12 decimal)
JNZ REPEAT ; If not zero (CL != 12), continue loop

INC DL ; Increment main digit pointer (DL = DL + 1)
CMP DL, C ; Compare DL with 12 decimal
JNZ START ; If not zero (DL != 12), restart main loop

END ; End of program

d) Displays number divided by 5: Ex: 00, 05, 10,….., 95

JMP start ; Skip the data section and jump to start label
; [Data section with 7-segment patterns omitted for brevity]
DB FB ;0
DB 0B ;1
DB B7 ;2
DB 9F ;3
DB 4F ;4
DB DD ;5
DB FD ;6
DB 8B ;7
DB FF ;8
DB DF ;9



start:
MOV AL, FA ; Load display control value (0xFA) into AL
OUT 02 ; Send control value to port 2 (display control port)
MOV CL, 03 ; Initialize counter to 3 (starting memory position)
LOOP:

```
MOV BL, 02      ; Initialize digit pointer BL to 2 (starting memory position)
start2:
; This block increments BL by 5 (via five INC instructions)
INC BL          ; Increment digit pointer (BL = BL + 1)
INC BL          ; Increment digit pointer (BL = BL + 1)
INC BL          ; Increment digit pointer (BL = BL + 1)
INC BL          ; Increment digit pointer (BL = BL + 1)
INC BL          ; Increment digit pointer (BL = BL + 1) [Total: BL += 5]
MOV AL, [BL]    ; Load digit pattern from memory at address BL
OUT 02          ; Output digit pattern to display port
CMP BL, 06      ; Compare BL with 6
JS start2       ; If BL < 6 (signed), jump back to start2
; The following section handles the tens digit
CMP CL, 0C      ; Compare CL with 12 (0x0C)
JNS end         ; If CL >= 12 (not signed), jump to end
JS start3       ; If CL < 12 (signed), jump to start3
start3:
MOV AL, [CL]    ; Load current digit pattern from memory at address CL
DEC AL          ; Decrement the digit pattern value (AL = AL - 1)
OUT 02          ; Output modified digit to display
MOV AL, FB      ; Load separator pattern (0xFB) into AL
OUT 02          ; Output separator to display
INC CL          ; Increment main counter (CL = CL + 1)
CMP CL, 0C      ; Compare CL with 12 again
JS LOOP         ; If CL < 12 (signed), jump to LOOP
JNS LOOP        ; If CL >= 12 (not signed), jump to LOOP (Note: redundant)
end:
HLT             ; Terminate program execution
```

