

Microcontroller Simulator

<http://softwareforeducation.com/electronics/notes/A2/sim/index.php>

IVec, PORTA, B & C TRISA, B & C, PRE, TMR

Example Code	Explanations																			
<pre> ; ===== ; ===== Example 1 ===== ; == This is Comment == ; ===== NOP NOP NOP NOP START: NOP NOP NOP NOP JMP START </pre>	<p>http://softwareforeducation.com/electronics/notes/A2/sim/01-nop-jmp.php</p> <p>ASSEMBLER This program translates human readable codes like NOP into binary understood by the microcontroller.</p> <p>NOP Do nothing for one clock cycle. It's used for small time delays. The six bit machine code for NOP is 00 0000₂ or 0x00.</p> <p>START: This labels address 0x4 (in this example)</p> <p>JMP START: Set the Program Counter PC to the specified address.</p> <p>PC 0x04 0000100</p> <p>The machine code for JMP is 10 1000₂ or 0x28.</p> <p>START: labels the destination address of 0x04.</p> <p>The full machine code is 0x2804.</p> <p>Here is the assembled machine code for this example program.</p> <p>The pink shading is the program counter pointing at address 0x08.</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td style="background-color: yellow;">0</td> <td style="background-color: yellow;">1</td> <td style="background-color: yellow;">2</td> <td style="background-color: yellow;">3</td> <td style="background-color: yellow;">4</td> <td style="background-color: yellow;">5</td> <td style="background-color: yellow;">6</td> <td style="background-color: yellow;">7</td> <td style="background-color: yellow;">8</td> </tr> <tr> <td style="background-color: yellow;">00</td> <td style="background-color: yellow;">0000</td> <td style="background-color: yellow;">0000</td> <td style="background-color: yellow;">0000</td> <td style="background-color: yellow;">0000</td> <td style="background-color: yellow;">0000</td> <td style="background-color: yellow;">0000</td> <td style="background-color: yellow;">0000</td> <td style="background-color: yellow;">0000</td> <td style="background-color: pink;">2804</td> </tr> </table> <p>To assemble the program, press A You must re-do this after every code change.</p> <p>To stop or single step the program, press S</p> <p>To run the program faster, press F or slower, press V</p>	0	1	2	3	4	5	6	7	8	00	0000	0000	0000	0000	0000	0000	0000	0000	2804
0	1	2	3	4	5	6	7	8												
00	0000	0000	0000	0000	0000	0000	0000	0000	2804											

<http://softwareforeducation.com/electronics/notes/A2/sim/03-Tlight.php>

```

; =====
; ===== Example 3 =====
; =====
MOVW 0x00
MOVWR TRISC

START:
MOVW 0x55
MOVWR PORTC

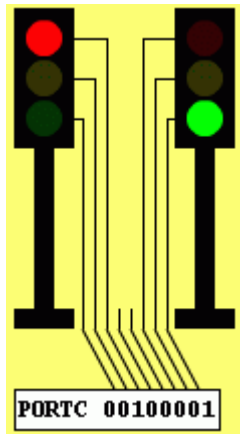
MOVW 0xA
MOVWR PORTC

MOVW 0xC
MOVWR PORTC

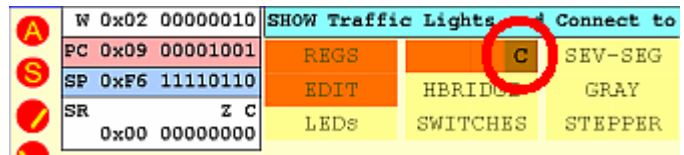
MOVW 0x2
MOVWR PORTC

JMP START

```



To connect the traffic lights to PORTC and make them visible, click here.

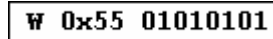


This example controls the traffic lights. The control data is incorrect and your task is to fix it.

MOVW 0x00
This will set all eight PORTC lines to outputs

MOVW TRISC
This copies W into TRISC to set the PORTC lines to outputs.

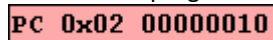
MOVW 0x55
This copies 0x55 into the working register



MOVWR PORTC This sends traffic lights control data to PORTC

START: This labels address 0x2.

JMP START This sets the program counter PC to 0x2



Here is the assembled machine code for this example.

	0	1	2	3	4	5	6	7	8	9	A
00	3000	01FC	3055	01FD	300A	01FD	300C	01FD	3002	01FD	2802

Use this grid to help design the correct traffic lights control data.

G	A	R	x	x	R	A	G	Hex	8	4	2	1	Hex
									0	0	0	0	0x0
									0	0	0	1	0x1
									0	0	1	0	0x2
									0	0	1	1	0x3
									0	1	0	0	0x4
									0	1	0	1	0x5
									0	1	1	0	0x6
									0	1	1	1	0x7
									1	0	0	0	0x8
									1	0	0	1	0x9
									1	0	1	0	0xA
									1	0	1	1	0xB
									1	1	0	0	0xC
									1	1	0	1	0xD
									1	1	1	0	0xE
									1	1	1	1	0xF

```

MOVW 0X1
MOVWR TRISA

LOOP:
MOVWR PORTA
ANDW 0x1
SUBW 0x1
JPZ TOOHOT
JMP TOOCOLD

TOOHOT:
; Add two lines here

JMP LOOP

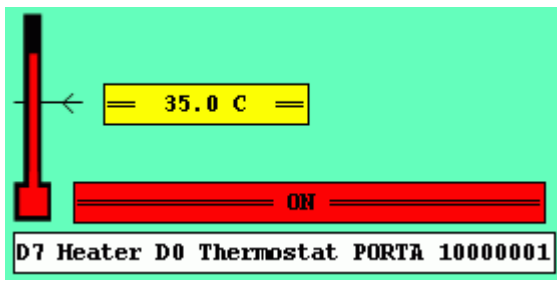
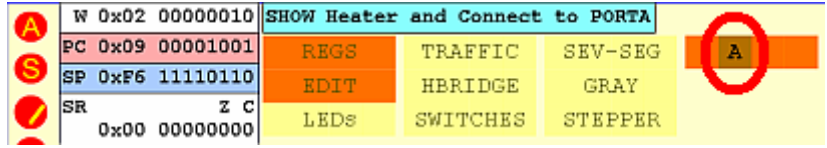
TOOCOLD:
; Add two lines here

JMP LOOP

```

<http://softwareforeducation.com/electronics/notes/A2/sim/03-Heater.php>

To connect the heater to PORTA and make it visible, click here.



Your task is to write code to make the temperature settle at about 22C.

```

MOVW 0x1
MOVWR TRISA

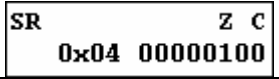
```

These two lines set bit 0 of PORTA to be an input.

MOVWR PORTA This line reads data from PORTA and copies it into W. Because this happens repeatedly, this is called polling PORTA.

ANDW 0x1 This is a bit mask. Bit 0 is unchanged and all the other bits are set to zero.

SUBW 0x1 Subtract 1 from W. If W contained one, it will now contain zero and the Z flag will have been set. You could alternatively use **XORW 0x1**



```

MOVW 0X00
MOVWR TRISC

START:
MOVW 0X21
MOVWR PORTC
MOVW 0X20
CALL DELAY

MOVW 0X62
MOVWR PORTC
MOVW 0X05
CALL DELAY

MOVW 0X84
MOVWR PORTC
MOVW 0X20
CALL DELAY

MOVW 0X46
MOVWR PORTC
MOVW 0X05
CALL DELAY

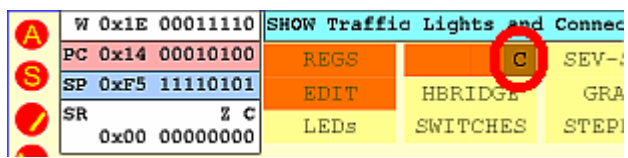
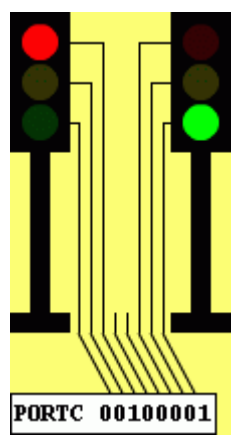
JMP START
; =====
DELAY:
SUBW 0X1
JPZ DONE
JMP DELAY

DONE:
RET

```

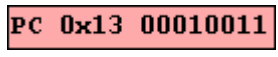
<http://softwareforeducation.com/electronics/notes/A2/sim/05-Delay.php>

Connect the traffic lights to PORTC

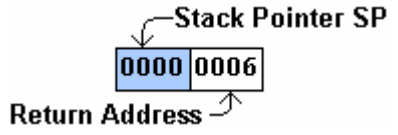


MOVW 0x20 0x20 controls the length of the time delay.

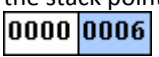
CALL DELAY Set PC to the start address of the delay subroutine.



Also the return address is saved onto the stack



RET When the subroutine returns, the return address is copied back into PC and the stack pointer is moved back to its original position.



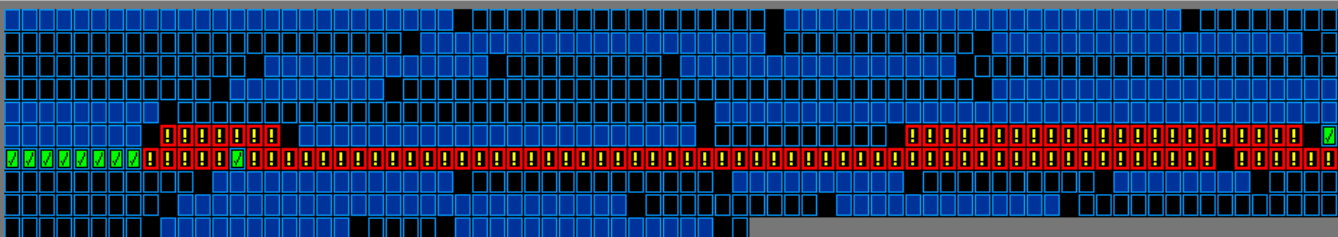
This subroutine code could be saved in a library for use in many different programs.

<pre> ; ===== ; Timer Interrupts ; ===== MOVW 0x1 MOVWR PRE MOVW TIMER MOVWR IVEC STI ; ===== ; loop should do ; something ; ===== START: NOP NOP NOP NOP JMP START ; ===== ; TIMER INTERRUPT ; HANDLER ; ===== TIMER: COUNT: MOVW 0x20 SUBW 0x1 JPZ DONE JMP COUNT DONE: RET ; ===== </pre>	<p>http://softwareforeducation.com/electronics/notes/A2/sim/07-Interrupts.php</p> <p>MOVW 0x1 Prescaler value MOVWR PRE Copy 0x1 to the prescaler. This will divide the clock by PRE + 1 MOVW TIMER TIMER labels the start address of the Timer Interrupt code. MOVWR IVEC Copy this start address into the IVEC register. Whenever the timer interrupt happens, this address will be used and the code at this address will be run. STI Set Timer Interrupt. Enable timer interrupt processing. Not in AQA spec'.</p> <p>Here is a do nothing or idle loop. The NOP commands make the loop execution visible and could be removed.</p> <p>When TMR is zero, the timer interrupt is triggered. The processor sets the PC to the address located in IVEC. The return address is saved onto the stack.</p> <p>When the interrupt is complete, the RET command causes the return address saved on the stack to be copied into PC so the processor resumes its previous task.</p> <p>TIMER: This labels the start address of the timer interrupt code. MOVW 0x20 This initialises W. Count down from 0x20 to zero. COUNT: This labels the point in this loop where the repetition will happen. SUBW 0x1 Subtract one from W and store the result in W. JPZ DONE This does a jump if the Z flag is set. The Z flag is set when a move or calculation leaves Zero in W. JMP COUNT Carry on counting down if the zero has not been detected. RET Return from the interrupt.</p>
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

<http://reviseomatic.enorf.ac.uk/v2/log/log-in.php>

reviseOmatic V2: All Topics Neil Bauers: enorf.ac.uk-2009-Test-Class

View: [All Selected Grades Help](#) Class: [Switch Join Leave](#) My Classes: [New Delete Rename](#) Topics: [View Add Remove](#) View: [Profile](#) [Logout](#)



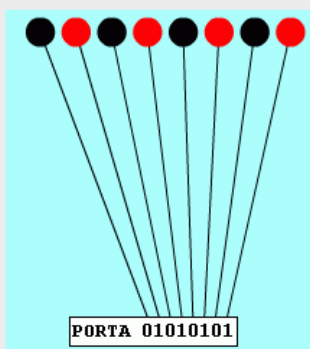
847 Correct

Click on the Help link and complete task (2) on the linked page.

If TRISA bits are set to ONE, the corresponding PORTA bits will be _____.

[Submit Answer](#) [Help](#)

Question 847 Correct.
23 A2 Assembly Code Programming
Assembler 2
Date: 2009-07-07
Attempts: 1 Correct: 1
Answer: C



reviseOmatic

- Approximately 400 AS test and revision questions.
- Approximately 400 A2 test and revision questions.
- Most questions have a help link to a relevant web page.
- Anyone can register.
- Anyone can set up a class and their friends can join their class. This is for student team work. In the same way, teachers can set up classes and their students can join.
- There are two views.
The “All Topics” view gives access to every AS and A2 question.
The “Selected Topic” view gives access to topics which have been added to a class.
Teachers or students should add topics to a class when the topic has been taught.
These questions are highlighted with an exclamation mark.
This indicates that a homework or personal deadline has been set and these questions should be prioritised.

Assembler language microcontroller instructions

Mnemonic	Operands	Description	Operation	Flags	Clock cycles
NOP	none	No operation	none	none	1
CALL	K	Call subroutine	stack \leq PC PC \leq K	none	2
RET	none	Return from subroutine	PC \leq stack	none	2
INC	R	Increments the contents of R	(R) \leq (R) + 1	Z	1
DEC	R	Decrements the contents of R	(R) \leq (R) - 1	Z	1
ADDW	K	Add K to W	W \leq W + K	Z, C	1
ANDW	K	AND K with W	W \leq W • K	Z, C	1
SUBW	K	Subtract K from W	W \leq W - K	Z, C	1
ORW	K	OR K and W	W \leq W + K	Z, C	1
XORW	K	XOR K and W	W \leq W \oplus K	Z, C	1
JMP	K	Jump to K (GOTO)	PC \leq K	none	2
JPZ	K	Jump to K on zero flag set	PC \leq K	none	2
JPC	K	Jump to K on carry flag set	PC \leq K	none	2
MOVWR	R	Move W to the contents of R	(R) \leq W	Z	1
MOVW	K, W	Move K to W	W \leq K	Z	1
MOVRW	R	Move the contents of R to W	W \leq (R)	Z	1

The red highlight may be a typo by AQA. For this line to make sense, it should read MOVW K