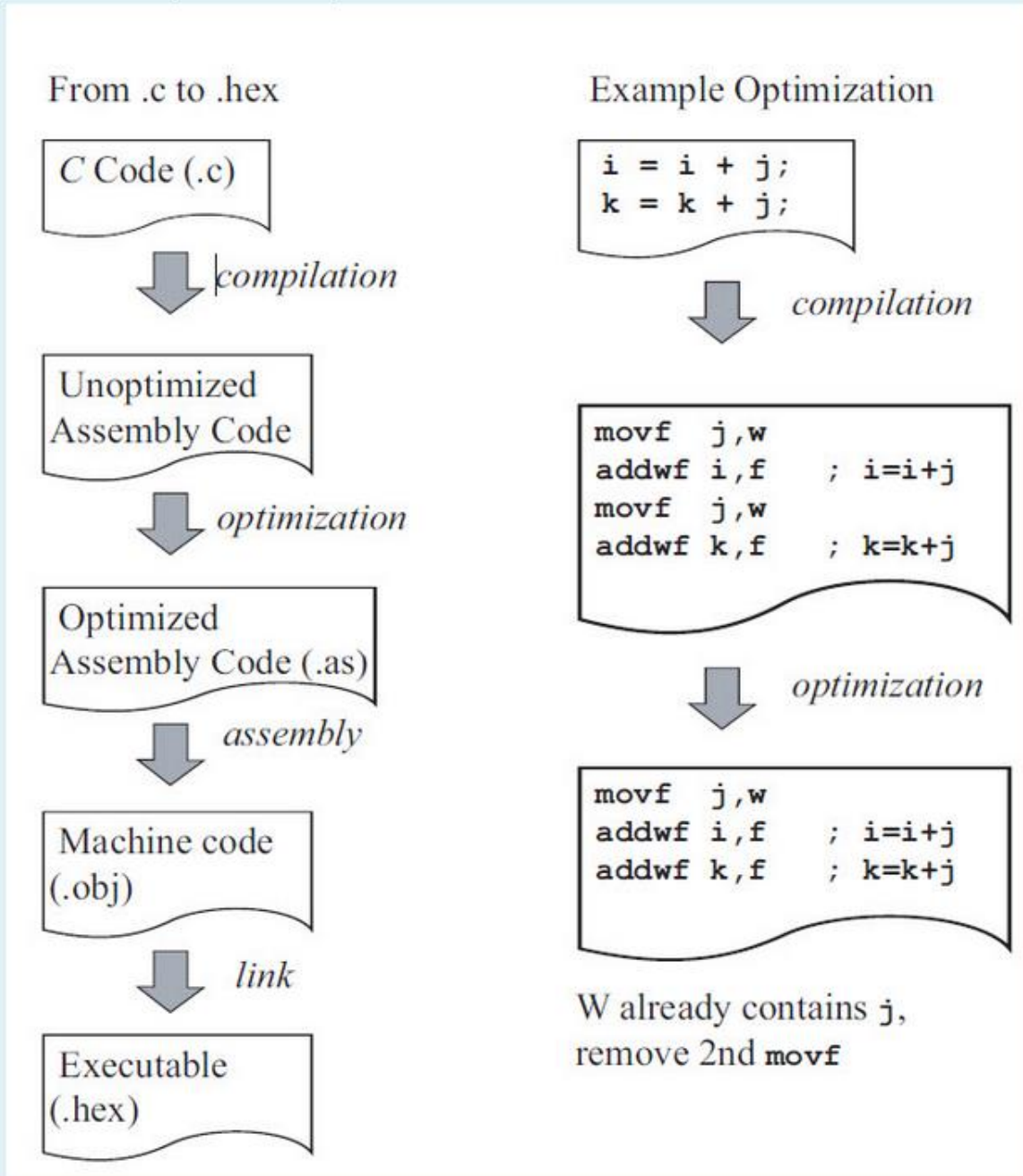## Questionnnaire Q9_12: Example questions referring to projects P9-P10-P11-P12

C compilers allows us programming microcontroller applications skipping the inner details and complications of assembly language. However, the optimisation process as represented for example in the picture below is not for free, excepting some demonstrative versions of the compilers, because optimising is not a simple process. Typically, code size is reduced and performance improved after optimisation.

From .c to .hex

C Code (.c)

⬇ compilation

Unoptimized Assembly Code

⬇ optimization

Optimized Assembly Code (.as)

⬇ assembly

Machine code (.obj)

⬇ link

Executable (.hex)

Example Optimization

```
i = i + j;
k = k + j;
```

⬇ compilation

```
movf   j,w
addwf  i,f    ;  i=i+j
movf   j,w
addwf  k,f    ;  k=k+j
```

⬇ optimization

```
movf   j,w
addwf  i,f    ;  i=i+j
addwf  k,f    ;  k=k+j
```

W already contains j, remove 2nd movf

Select one:

Answers

◉ True

○ False

These are common logic bitwise operations coded in both, C and assembly languages. Name them.

In C	In Assembly	Execution

i = i & 0x0F;
 	
 A)
	```movf 0x020,w```
	```andlw 0x0F```
	```movwf 0x20```
		```i = 0x2C = 0010 1100```
		``` &&&& &&&&```
		```mask = 0x0F = 0000 1111```
		``` ---------```
		```result = 0000 1100 = 0x0C```

i = i | 0x0F;
 	
 B)
	```movf 0x020,w```
	```iorlw 0x0F```
	```movwf 0x20```
		```i = 0x2C = 0010 1100```
		``` |||| ||||```
		```mask = 0x0F = 0000 1111```
		``` ---------```
		```result = 0010 1111 = 0x2F```

i = i ^ 0x0F;
 	
 C)
	```movf 0x020,w```
	```xorlw 0x0F```
	```movwf 0x20```
		```i = 0x2C = 0010 1100```
		``` ^^^^ ^^^^```
		```mask = 0x0F = 0000 1111```
		``` ---------```
		```result = 0010 0011 = 0x23```

i = ~i;
 	
 D)
	```comf 0x020,f```	```i = 0x2C = 0010 1100```
		``` ~~~~ ~~~~```
		```result = 1101 0011 = 0xD3```

Select one:

- ○ 1.   A) AND;   B) NOR;   C) NXOR;   D) NOT
- ⦿ 2.   A) AND;   B) OR;   C) XOR;   D) NOT
- ○ 3.   A) OR;   B) AND;   C) NOT;   D) XOR
- ○ 4.   A) AND;   B) OR;   C) NOT;   D) XOR

Microcontrollers are characterized by having small amounts of program (flash memory) and data (RAM) memory, and take advantage of the Harvard architecture to speed processing by concurrent instruction and data access. The separate storage means the program and data memories may feature different bit widths, for example using 16-bit wide instructions and 8-bit wide data.

This is a simplified RTL schematic of a PIC CPU from Microchip. Which are the RAM and program memory addressing capacities?
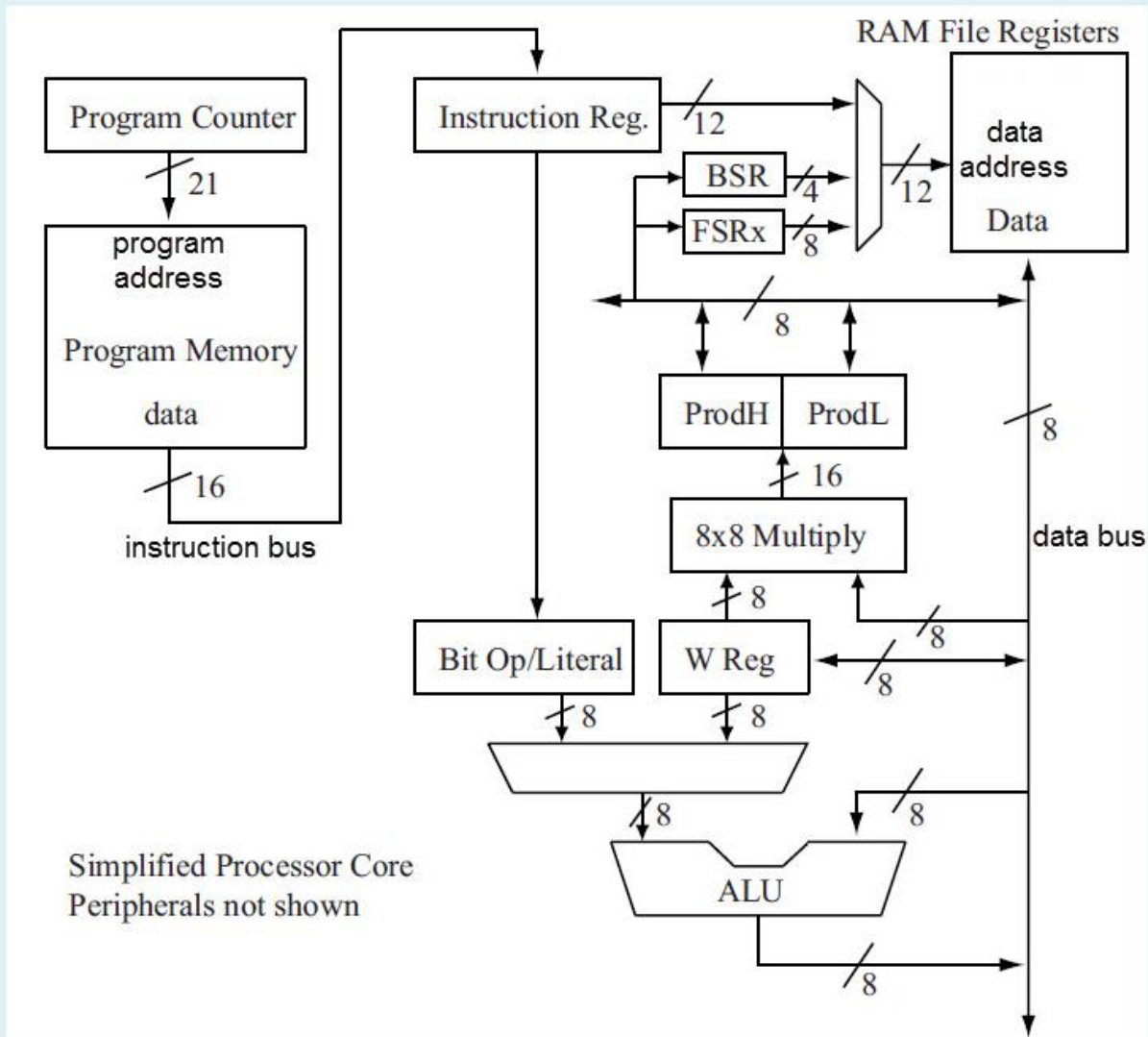
Select one:

- ⦿ a. RAM memory = 4 kB (kbytes); Program memory = 2 M positions (16-bit words)

- ○ b. RAM memory = 512 B (bytes); Program memory = 512 k positions (16-bit words)

- ○ c. RAM memory = 1 kB (kbytes); Program memory = 1 M positions (16-bit words)

- ○ d. RAM memory = 4 kB (kbytes); Program memory = 64 k positions (16-bit words)

This application uses the LCD board LM032L based on Hitachi HD44780 controller chip to represent ASCII messages. The code is compiled using a high level library of functions in C language, and it takes several ms to write all the characters. Which is the incorrect statement?
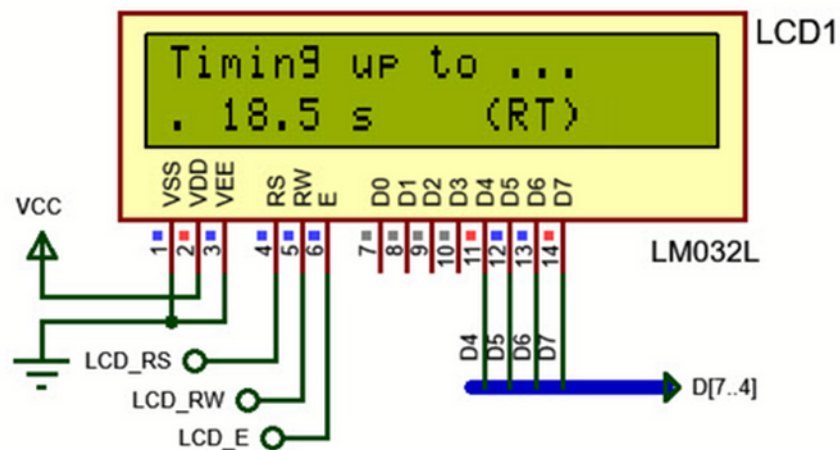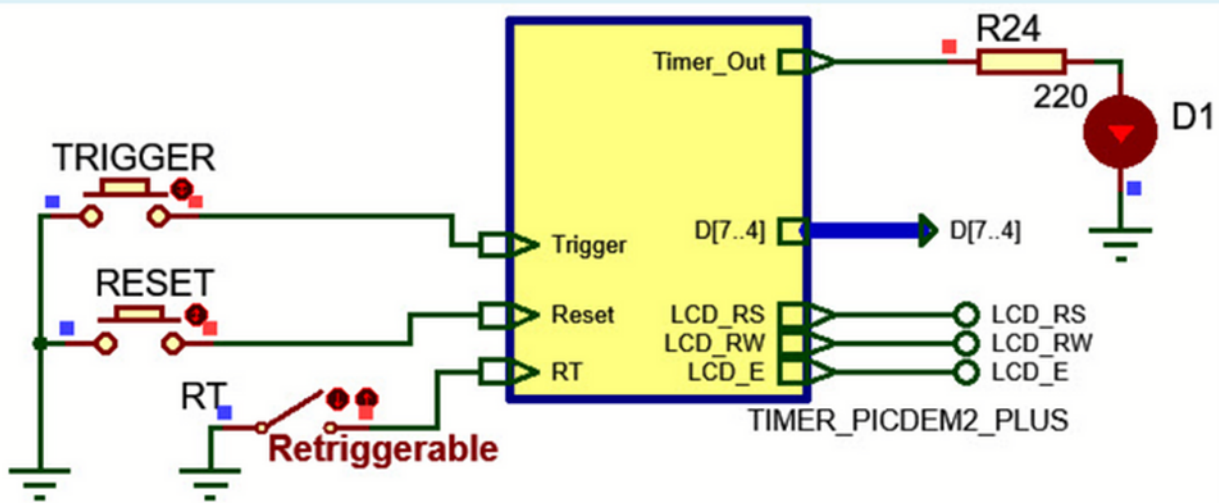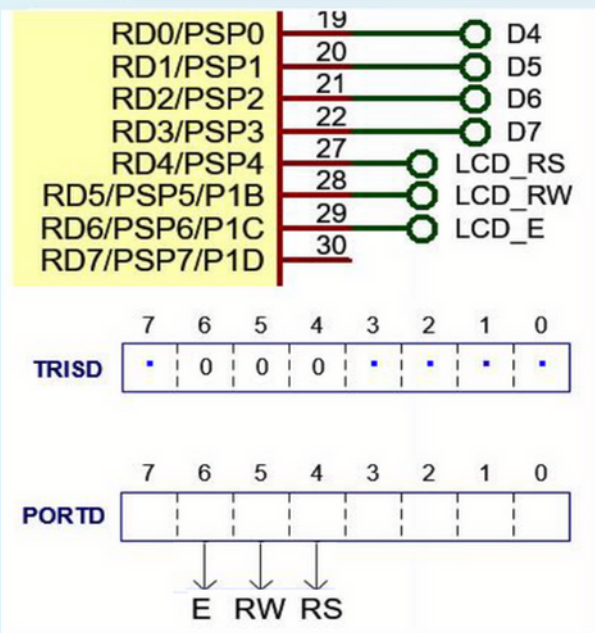

Select one:

- ◉ a. To use the LCD display efficiently, we will include the code below to write messages in the *output_logic()* function and it will be executed in every loop:

    lcd_clear();

    lcd_home();

    lcd_puts("Timing up to ...");

    lcd_home2();

    if (Value_RT == 1) lcd_puts (". 18.5 s   (RT)");

    else lcd_puts(". 18.5 s   (nRT)")

- ○ b. To compile the project without errors when using LCD functions, header files (.h) and LCD functions source files (.c) must also be included.

In order to configure the LCD hardware, the control lines RS, E, RW must be set as outputs:

RD0/PSP0	19		D4
RD1/PSP1	20		D5
RD2/PSP2	21		D6
RD3/PSP3	22		D7
RD4/PSP4	27		LCD_RS
RD5/PSP5/P1B	28		LCD_RW
RD6/PSP6/P1C	29		LCD_E
RD7/PSP7/P1D	30		

	7	6	5	4	3	2	1	0
TRISD	.	0	0	0	.	.	.	.

	7	6	5	4	3	2	1	0
PORTD								

E RW RS

Signal	I/O	Device Interfaced with	Function (Pin Functions)
RS	I	MPU	Selects registers. 0: Instruction register (for write) Busy flag: address counter (for read) 1: Data register (for write and read)
R/W̄	I	MPU	Selects read or write. 0: Write 1: Read
E	I	MPU	Starts data read/write.
DB4 to DB7	I/O	MPU	Four high order bidirectional tristate data bus pins. Used for data transfer and receive between the MPU and the HD44780U. DB7 can be used as a busy flag.
DB0 to DB3	I/O	MPU	Four low order bidirectional tristate data bus pins. Used for data transfer and receive between the MPU and the HD44780U. These pins are not used during 4-bit operation.

The *char* type variable *var_LCD_Flag* will be used to control when to write messages in the LCD: this flag will be set only when there is new information to represent on the display.

This is the PIC18F4520 Timer0 (**TMR0L**) block diagram for 8-bit mode of operation. We program the configuration bits as follows in order to use the Timer0 to measure the frequency of an external signal connected to the **T0CKI** pin:

T0SE = 0; **T0CS** = 1; **PSA** = 0; **TMR0L** = 0x00; **T0PS**(2..0) = "011"; and the Fosc = 4 MHz.

If we enable by software the Timer0 for **100 ms**, which is the maximum frequency that can be measured before there is an overflow error that invalidates the measurement?
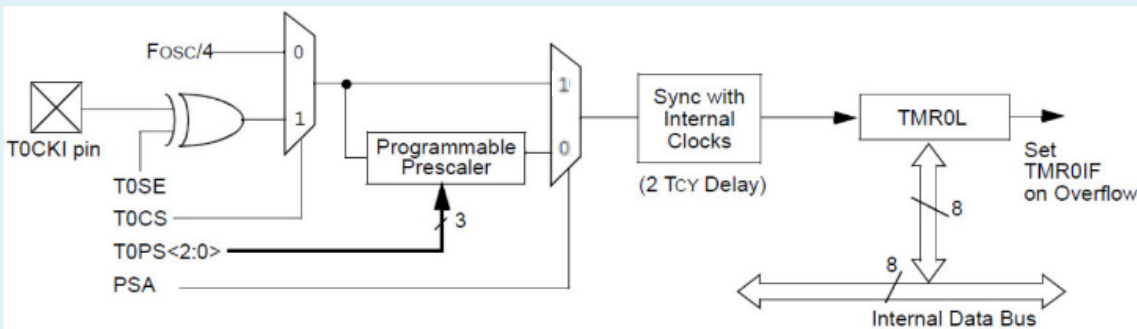
T0PS<2:0>: Timer0 Prescaler Select bits

111 = 1:256 Prescale value
110 = 1:128 Prescale value
101 = 1:64 Prescale value
100 = 1:32 Prescale value
011 = 1:16 Prescale value
010 = 1:8   Prescale value
001 = 1:4   Prescale value
000 = 1:2   Prescale value

Select one:

○ a. $F_{max} = 250\ Hz$

○ b. $F_{max} = 4.096\ Hz$

○ c. $F_{max} = 250\ kHz$

◉ d. $F_{max} = 40.96\ kHz$