Anthony Debruyn
Quentin Delhaye
Alexis Lefebvre
Aurélien Plisnier

Labo 5

dsPIC33

Dragomir Milojevic & Yannick Allard ELEC-H-473

1 Question 1

dsPIC30F-33F CPU reference manual p 2-2:
"The dsPIC33F/PIC24H CPU has a 16-bit (data) modified Harvard architecture with an enhanced instruction set ... The core has a 24-bit instruction word, with a variable length opcode field ... A single-cycle instruction prefetch mechanism helps maintain throughput and provides predictable execution. All instructions execute in a single cycle, except the instructions that change the program flow."

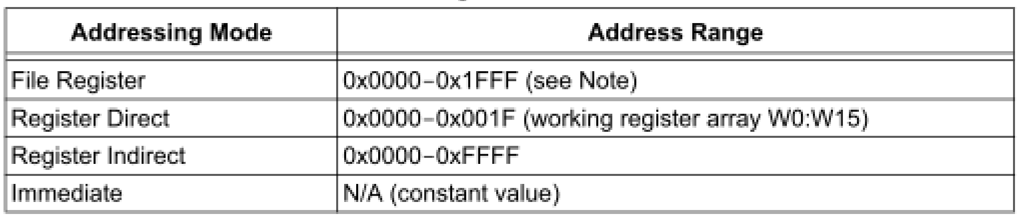
2 Question 2

[1] "Microprocessor is an IC which has only the CPU inside it i.e. only the processing powers.... These microprocessors don't have RAM, ROM, and other peripheral on the chip.... Microcontroller has a CPU, in addition with a fixed amount of RAM, ROM and other peripherals all embedded on a single chip... Microcontrollers are designed to perform specific tasks... Microprocessor find applications where tasks are unspecific."

3 Question 3

Programmer's Reference Manual, page 4-2.

Table 4-1: dsPIC30F/33F Addressing Modes



Addressing Mode	Address Range
File Register	0x0000–0x1FFF (see Note)
Register Direct	0x0000–0x001F (working register array W0:W15)
Register Indirect	0x0000–0xFFFF
Immediate	N/A (constant value)

Note: The address range for the File Register MOV is 0x0000–0xFFFE.

4 Question 4

4.1 File Register

Programmer's Reference Manual, page 4-2.
"File register addressing is used by instructions which use a predetermined data address as an operand for the instruction.
The MOV instruction provides access to all 64 Kbytes of memory using file register addressing. The MOV instruction accesses all 64K of memory as words."

4.2 Register Direct

Programmer's Reference Manual, page 4-4.
"Register direct addressing provides the ability for dynamic flow control. Since variants of the DO and REPEAT instruction support register direct addressing, flexible looping constructs may be generated using these instructions."

4.3 Register Indirect

Programmer's Reference Manual, page 4-5.
"Register indirect addressing is used to access any location in data memory by treating the contents of a working register as an Effective Address (EA) to data memory.
The MOV with offset instructions provides a literal addressing offset ability to be used with indirect

addressing. In these instructions, the effective address is formed by adding the contents of a working register to a signed 10-bit literal."

4.4 Immediate

Programmer's Reference Manual, page 4-9.
"In immediate addressing, the instruction encoding contains a predefined constant operand."
... C'est l'histoire d'un pingouin qui respire par le cul: il s'assied il meurt.

5 Question 5

Programmer's Reference Manual, page 1-2.
"dsPIC30F/33F: 24-bit Instruction Word length, 16-bit Data Path" In page 2-2 we read: "the core has a 24-bit instruction word" so it seems coherent.

6 Question 6

Since the instruction is encoded in a limited number of bits, the maximum size of the associated constant is limited.

Question 7

The header file is associated to the studied processor. It defines the registers to use for this specific processor, and the ports to use. If we wanted to use a different processor, we would just need to replace the ".h" file by the corresponding one.

Question 8

Declaring a variable volatile tells the compiler that the variable may be changed at any moment by an external intervenant. Knowing that, the compiler does not optimize the parts using the variable and does not use cached version of it.

Question 9

Type	Min	Max
char, signed char	-128	127
unsigned char	0	255
short, signed short	-32768	32767
unsigned short	0	65535
int, signed int	-32768	32767
unsigned int	0	65535
long, signed long	-2147483648	2147483647
unsigned long	0	4294967295
long long**, signed long long**	-9223372036854775808	9223372036854775807
unsigned long long**	0	18446744073709551615
float	1.175e-38	3.403e38
double*	1.175e-38	3.403e38
long double	2,22507385850720138309023...e-308	1,7976931348623159077293...e+308

Question 10

Based on the table 5-1 of [2], we see that signed and unsigned type are the same. Hence, by default the types are signed.

Question 11

It allows the representation of signed and unsigned integers on 8, 16, 32 and 64 bits by using the different types. It does the same for floating point in 32 and 64 bits. The type sizes can vary for different processors, so it is better to define new data types, used in the program, that always have the same size. It also removes the ambiguity around the default signed/unsigned type status.

Question 12

A logician's wife is having a baby. The doctor immediatly hands the newborn to the father. His wife asks impatiently: "So, is it a boy or a girl?" The logician replies: "Yes."

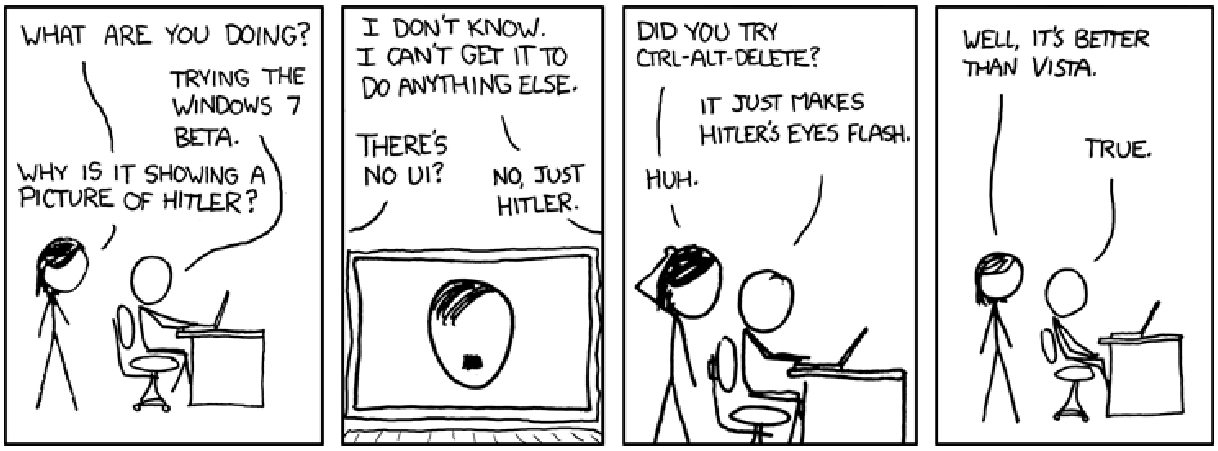
Question 13

Variable	Storage address
a	0831
b	0830
c	082E
d	082C
e	0828
f	0824
g	081C
h	0814
i	0810
j	0808
glob1	0010

Question 14

All the variables are initialized with integer constants. All the variables are addressed absolutely.

Question 15


WHAT ARE YOU DOING?
TRYING THE WINDOWS 7 BETA.
WHY IS IT SHOWING A PICTURE OF HITLER?
I DON'T KNOW. I CAN'T GET IT TO DO ANYTHING ELSE.
THERE'S NO UI?
NO, JUST HITLER.
DID YOU TRY CTRL-ALT-DELETE?
IT JUST MAKES HITLER'S EYES FLASH.
HUH.
WELL, IT'S BETTER THAN VISTA.
TRUE.

Question 16

The 64 bits variables aren't seem to be correctly displayed in the watch. It seems to keep only the 32 low order bits.

Variable	Declaration	Memory
g	`0xFEDCBA9876543210`	`0x76543210`
h	`-0x123456789ABCDEF0`	`0x76543210`
j	8.76543E-18	543029197436e-18

Question 17

The `INT64S` variable `h` has been declared has equal to `-0x123456789ABCDEF0`, but is stored in the memory has `0xFEDCBA9876543210`. The `FP64` variable `j` has been declared has equal to `8.76543E-18` (which is `0x3C64363254B46D0F`, but is stored in the memory has `0x3C64363260000000`.

Question 18

The relevant warnings are those about the `long` overflow. Howerver, the variables concerned by those warnings are declared as `long long`, which is enough to store the 64 bits integer constant of their declaration. Those warnings are thus not justified.

Bear in mind that the others are (unused variables, etc.), but are not relevant for the execution of this program.

Question 19

`glob1` is declared outside of the `main` and initialized at the end of the `main`.

Question 20

`register` variable_type variable_name `asm ("`register_name`")`

Question 21

As the warnings state, the declared values are to great for the variables type, and are thus truncated. However, the signed variables generate an overflow exception. In that case, the compiler considers the variable as unsigned and then truncates it.

Question 22

The initializations are made through the value of the other variables, plus they are all global.

Question 23

In the disassembly listing, the assignations work in the reverse order. Instead of having `opcode destination, source`, we have `opcode source, destination`.

Suggestions

- Prepare the project files so that they're directly usable with MPLAB X. MPLAB 8.92 is great, but not usable under Linux/Mac without emulation or virtualization.
- Add more questions.

References

[1] http://www.engineersgarage.com/tutorials/difference-between-microprocessor-and-microcontroller

[2] Microchip Technology Inc., *MPLAB® C30 C COMPILER USER'S GUIDE*, 2007.