Hardwarenahe Programmierung

8051 Project 1: Arithmetic units

1 Project's goal

The goal is to develop an assembly programs capable of performing basic arithmetic operations on data larger than one byte (8 bits).

2 Keil μVision quick start

2.1 Opening projects

To open the project file `add32`:

- Start Keil μVision3 from the Windows Start menu.
- Make sure that no project is open. Select *Project→Close Project*.
- Open the project add32:
  - select *Project→Open Project...*,
  - navigate to where you unpacked the project zip-file and then to `project 1 - arithmetic units\add32`,
  - select the file `add32.uv2`.

2.2 Saving files

To save the currently selected text file (the one shown in the editor) select *Project→Save*, or hit Ctrl+S.

2.3 Assembling and building

Open a project. Make sure that you are not in Debug Mode (see below). Select *Project→Build target*. Analyze and correct any errors, then re-build.

2.4 Starting debug session/analyzing programs

After successful build of your project select *Debug→Start/Stop Debug Session*. The screen changes and you can see the register window to the left. In this mode you can use the *Debug* menu to run or step through the program. You can also set breakpoints at which the execution will halt automatically. The commands are also available in the toolbar directly above the register window, and through keyboard shortcuts (see the *Debug* menu).

Note: You have to leave the Debug Session before you can re-build your project.

2.5 Displaying memory contents

Select *View→Memory window* to toggle the visibility of the memory window. In the *Address* box of this widow enter:

- `d:0` to see the contents of the internal RAM,
- `x:1234h` to see the contents of the external RAM starting from the address 01234h.

Hint: To get accustomed to the program you can load one of the demo programs shown in the lecture (you can download them from the institute's website) and experiment with the features described above.

Hint: Do not hesitate to refer to the on-line help of the Keil μVision software for additional information on assembler statements and 8051 instructions. In particular take advantage of the chapter *Instructions* in the book *8051 Instruction Set*.

3 Addition

Project name: add32 – Create an assembly program which performs addition of two 32-bit wide numbers.

3.1 Fixed address

Create an initial version which:

- uses addresses 40h to 43h, as well as 50h to 53h as summands for the addition (43h and 53h are the least significant bytes (LSBs)),
- stores the result in the internal RAM at the address of the first summand, i. e. 40h to 43h with LSB again in 43h,

- define the addresses using the assembler statement `EQU`; follow the example given in the project file,
- remember that addition can generate carry signals – take them into account.

3.2 Variable addresses

Modify the program from Section 3.1 such that:

- the summands are taken from 4 consecutive bytes ending at addresses specified by the contents of registers R0 and R1, respectively(i.e. R0 and R1 contain the address of the LSB),
- all addition operations are performed in a loop.

Hint: Use the indirect addressing mode for the `ADDC` instruction. Consider the following code:

```
MOV R0, #3Fh
ADDC A, @R0
```

It adds the contents of the accumulator to the contents of the cell at address specified by the contents of the register R0 - in this case 3fh.

3.3 Packaging

Define a subroutine `ADD32`, and put the contents of your program inside. Write the main program which will call the subroutine to add numbers 0FEEDBEEFh and 0ABBADEADh stored starting from addresses 33h and 66h respectively. Check if the contents of cells 30h-33h is 0AAA89D9Ch after the addition. Is this the correct result?

4 Multiplication

Project name: mul16 – Write an assembly program for multiplying two 16-bit wide numbers. The result shall be a 32-bit wide number.

4.1 Multiplication of 8-bit numbers

The 8051 has the instruction `MUL AB`, which multiplies the contents of the accumulator with the contents of the register B. The result is a 16-bit number, whose MSB is delivered to B, and the LSB is stored in the accumulator.

```
MOV A, #05
MOV B, #05
MUL AB
```

After the execution of the above code the accumulator contains 19h (=25), the register B contains 0.

4.2 16-bit multiplication using the 8-bit `MUL AB` instruction

Take advantage of the follwing property of multiplication to be able to use the 8-bit mutliplication instruction `MUL AB` to compute the product of 16-bit wide numbers:

Let Z be a 16-bit wide number, and A_1 and A_0 its 8-bit components such that:

$$Z = A_1 \cdot 2^8 + A_0 \cdot 2^0 = A_1 A_0.$$

Then for the multiplcation of of 16-bit wide numbers Z and Y, $Z = A_1 A_0$, $Y = B_1 B_0$ we have:

$$(A_1 \cdot 2^8 + A_0 \cdot 2^0) \cdot (B_1 \cdot 2^8 + B_0 \cdot 2^0) = A_1 \cdot B_1 \cdot 2^{16} + A_1 \cdot B_0 \cdot 2^8 + A_0 \cdot B_1 \cdot 2^8 + A_0 \cdot B_0 \cdot 2^0.$$

The partial products $A_i \cdot B_j$ can be computed using 8-bit multiplication.

Write an assembly program which:

- multiplies the 16-bit numbers stored at fixed locations A1,A0 and B1,B0 (use the `EQU` assembler statement to define them),
- stores the 32-bit result in fixed location E3,E2,E1,E0 (again make use of `EQU`).

Note: For both multiplicands and the result use the *big endian* byte ordering, i. e. more significant bytes are stored at lower addresses in the memory.

4.3 `MUL16` subroutine

Modify the program from Section 4.2 such that:

- the 16-bit multiplication procedure is contained in the subroutine `MUL16`,
- the subroutine takes the address of LSBs of both multiplicands A and B (i. e. A0 and B0) from R0 and R1, respectively,
- the subroutine stores the result in the fixed location E3,E2,E1,E0 (no change compared to Section 4.2).

5 Average

Project name: average – Given four 16-bit numbers compute the integer average of their squares:

- sum all squares first, then compute the average; take care not to introduce errors due to truncation,
- provide sufficient memory space for the sum (how many bits are required to store the sum of 2^n 32-bit numbers?),
- re-use the code created in Section 3 and Section 4 (subroutines `ADD32` and `MUL16`),
- do not use the `DIV` instruction to compute the average,
- only the integer part of the average is of interest.

Hints:

- Division by two can be performed by using the `RRC` instruction. Refer to the Keil μVision on-line help for details.
- Use the `LOW` and `HIGH` assembler operators to extract the LSB and MSB, respectively, from 16-bit constants defined using `EQU`.

6 Example data for testing

This is some sample reference data for testing your programs. Note that the **result** column shows the contents of the result memory, not necessarily the correct mathematical result.

operation	**op1**	**op2**	**result**
ADD32	011223344h	0ccddeeffh	0de002243h
	0feedbeefh	0abbadeadh	0aaa89d9ch
	0fedcba98h	001234567h	0ffffffffh
	0dcdeffffh	0edecffffh	0cacbfffeh
MUL16	0baabh	0efbeh	0aed02feah
	0abbah	0beefh	080145ea6h
	0ffffh	0ffffh	0fffe0001h
	0f0e1h	0d2c3h	0c6500d63h
operation	**op1/op3**	**op2/op4**	**result**
average	08899h	0aabbh	
	0ccddh	0eeffh	08f73fa35h
	0deadh	0beefh	
	0feedh	0abbah	0b04953a5h