Oprea Sergiu-Daniel, Gr. 30322/2, Homework Lab4

1. Answer the following questions:
2. Explain how the following shift and rotate instructions work:
3. SHL, SAL, SHR, SAR



1. ROR, ROL, RCR, RCL
2. These instructions perform right or left rotations of the operand, based on the number of binary positions. The difference between the previous shift operations is the fact that the freed binary position is added to the beginning or the end of the binary number (depending on the left or right rotation). RCL and RCR perform the rotation with the involvement of the carry flag CF, while ROR and ROL don’t include this flag.

D7 D6 .... D1 D0 CF RCR

D7 D6 .... D1 D0 CF ROR

Instruction syntax:

ROR <parameter\_1>, <parameter\_2>

ROL <parameter\_1>, <parameter\_2>

RCR <parameter\_1>, <parameter\_2>

RCL <parameter\_1>, <parameter\_2>

1. Explain how the following jump instructions work:
2. CALL, RET

The CALL instruction performs a jump to the specified address in the instruction. Before the jump, the CPU saves on the program stack the return address. The RET instruction placed at the end of the routine performs the return in the appellant program. Hence, it extracts from the program stack the return address and makes the jump to it.

Instruction syntax:

CALL <address>

RET [<constant>]

where:

<address> - label with the name of the routine

<constant> - indicates the number of positions it must download from the stack before returning from the routine; this parameter usually is missing.

1. JMP

These instructions modify the execution sequence of instructions. There are unconditional jump instructions, which execute in any condition, and conditional jumps which execute when a specific condition is fulfilled (such as a flag or combination of flags).

Instruction syntax:

JMP <address>

J<cc> <address>

where:

<address> - label

<cc> - combination of letters that imply the condition

1. LOOP

These instructions allow the implementation of control structures similar to “for”, “while”, “do-until” from higher level programing languages. These instructions execute the following sequence of instructions: decrement the CX register (this is used as a counter), test for the terminating condition and jump to label (declared at the beginning of the loop sequence) if the test condition is not fulfilled.

Instructions syntax:

LOOP <address>

LOOPZ <address>

LOOPNZ <address>

where: <address> is a label declared by the programmer in the source code.

Instruction LOOP has CX=0 as the terminating condition (counter reaches 0), while LOOPZ tests for ZF=0 and LOOPNZ tests for ZF=1.

1. JC, JE, JZ, JS

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Instruction | Condition | Equivalent instructions | Explanations | Tested Flags |
| JC | CF=1 | JB,JNAE | Jump if carry was used |
| JS | SF=1 |  | Jump is result is negative |
| JZ | ZF=1 | JE | Jump if result is zero |
| JE | = | JZ | Jump if equal | ZF=1 |

1. JA, JB

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Instruction | Condition | Tested flags | Equivalent instructions | Explanations |
| JA | > | CF=0,ZF=0 | JNBE | Jump if greater |
| JB | < | CF=1 | JNAE,JC | Jump if lower |

1. JG, JL

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Instruction | Condition | Tested flags | Equivalent instructions | Explanations |
| JG | > | SF=OF sau ZF=0 | JNLE | Jump if greater |
| JL | < | SF!=OF | JNGE | Jump if lower |

1. Explain how the following I/O instructions work: IN, OUT

These instructions are used for transferring data with the registers (ports) of the in/out interface. Note that these are the only instructions in Intel CPUs that utilize ports.

IN & OUT instructions :

IN <accumulator>, <port\_address>

OUT < port\_address >, <accumulator>

where:

<accumulator> - register AX for 16 bit transfer or AL for 8 bit transfer

<port\_address> - address expressed on 8 bit or register DX

1. Explain how the following string instructions work:
2. MOVSB, MOVSW

These instructions transfer one element from the source array to the destination array. MOVSB works on byte (8 bit), while MOVSW on word (16 bit). The second instruction increments or decrements the index registers by 2, because each element of the array occupies 2 bytes. Note that these instructions don’t have parameters.

Example:

mov esi, offset source\_array ; "offset" is a keyword that determines the offset address of the variable

mov edi, offset destination\_array

mov ecx, array\_length

et: MOVSB ; DS:[SI]=>ES:[DI], SI++, DI++, CX--

jnz et

1. LODSB, LODSW, STOSB şi STOSW

The first 2 instructions perform successive load of the elements of an array in the accumulator register. The next 2 instruction perform the reverse operation (store the accumulator register in an array). These instructions also use the index registers (ESI for load and EDI for store), they are decremented or incremented automatically, while register CX is decremented. The letter “B” or “W” at the end indicate byte or word (8 or 16 bit transfer).

1. REP

These instructions allow the multiple execution of an instruction on an array. By using any of these instructions in front of an array instruction, it forces the CPU to repeat the operations until a terminating condition is fulfilled. The first instruction REP has CX=0 as the terminating condition. REPZ and REPE allow the operation to repeat until the result is 0 or the operands are equal. For REPNZ and REPNE, the operation is repeated as long as the result is not 0 or the operands are different.