Computer Organization &
Assembly Language Programming

CSE 2312
Lecture 22 Registers

Junzhou Huang, Ph.D.
Department of Computer Science and Engineering
Registers

• **Registers in the 8088 Processor**
  - Every processor, including the 8088, has an internal state, where it keeps certain crucial information.
  - Thus, the processor has a set of registers where this information can be stored and processed.
  - Probably the most important of these is the PC (program counter), which contains the memory location, that is, the address, of the next instruction to be executed.
  - This register is also called IP (Instruction Pointer). This instruction is located in a part of the main memory, called the code segment.

• **Different Registers**
  - General registers
  - Segment registers
  - Pointer and index registers
  - Condition code register or Flag register
  - Instruction pointers, program counter
# General Registers

<table>
<thead>
<tr>
<th>General registers</th>
<th>Segment registers</th>
</tr>
</thead>
<tbody>
<tr>
<td>AX, BH, CH, DH</td>
<td>CS, DS, SS, ES</td>
</tr>
<tr>
<td>AH, BL, CL, DL</td>
<td>Code segment</td>
</tr>
<tr>
<td></td>
<td>Data segment</td>
</tr>
<tr>
<td></td>
<td>Stack segment</td>
</tr>
<tr>
<td></td>
<td>Extra segment</td>
</tr>
</tbody>
</table>

### Pointer and index

<table>
<thead>
<tr>
<th>SP, BP, SI, DI</th>
<th>SF, CC, OD, TS, A, P, C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stack pointer</td>
<td>Status flags</td>
</tr>
<tr>
<td>Base pointer</td>
<td></td>
</tr>
<tr>
<td>Source index</td>
<td></td>
</tr>
<tr>
<td>Destination index</td>
<td></td>
</tr>
</tbody>
</table>

### Condition codes

<table>
<thead>
<tr>
<th>SF</th>
<th>O, D, I, T, S, Z, A, P, C</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>Status flags</td>
</tr>
<tr>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>IP</th>
<th>PC</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Instruction pointer</th>
<th>Program counter</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>0</td>
</tr>
</tbody>
</table>
General Registers (1)

• Several different general registers
  – The registers AX, BX, CX, and DX are the general registers.
  – The registers AH, AL, BH, BL, CH, CL, DH, DL are the general registers.

• Accumulator Register AX
  – AX is one of the general registers, which is called the accumulator register.
  – Used to collect results of computations
  – The target of many of the instructions.
  – AX is often the implied destination, for example, in multiplication.

• Base Register BX
  – BX is the base register.
  – For many purposes BX can be used in the same way as AX, but BX can hold a pointer to memory, AX cannot.
  – MOV AX,BX which copies to AX the contents of BX. (BX contains the source operand)
  – MOV AX,(BX) which copies to AX the contents of the memory word whose address is contained in BX. (BX points to the source operand)
General Registers (2)

• **Counter Registers CX**
  – Besides fulfilling many other tasks, this register is specifically used to contain counters for loops.
  – It is automatically decremented in the LOOP instruction, and loops are usually terminated when CX reaches zero.

• **Data Register DX**
  – It is used together with AX in double word length (i.e., 32-bit) instructions.
  – In this case, DX contains the high-order 16 bits and AX contains the low-order 16 bits. Usually, 32-bit integers are indicated by the term long.

• **Eight 8-bit Registers**
  – All of these general registers can be regarded either as a 16-bit register or as a pair of 8-bit registers. AX can be regarded as AH and AL.
  – In general, instructions doing arithmetic use the full 16-bit registers, but instructions dealing with characters usually use the 8-bit registers.
  – AL and AH are just names for both halves of AX. When AX is loaded with a new value, both AL and AH are changed to the lower and upper halves of the 16-bit number put in AX, respectively.
Example

• A Simple Example
  – What is the value in AX after executing the following instructions, where 258 is a decimal number?

    MOV  AX, 258
    ADDB AH, AL

• Solution
  – \((258)_{10} = (0000\ 0001\ 0000\ 0010)_{2}\)
  – MOV  AX, 258  \(\rightarrow\) AX= \((0000\ 0001\ 0000\ 0010)_{2}\)
  – \(\rightarrow\) AH= \((0000\ 0001)_{2}\)  AL = \((0000\ 0010)_{2}\)
  – ADDB AH, AL  \(\rightarrow\) AH= \((0000\ 0011)_{2}\)
  – \(\rightarrow\) AX= \((0000\ 0011\ 0000\ 0010)_{2} = (770)_{10}\)
Exercise

• A
  – What is the value in AX after executing the following instructions, where 256 is a decimal number?

  MOV AX, 256
  ADDB AH, AL

• Solution
  – \((256)_{10} = (0000 0001 0000 0000)_{2}\)
  – MOV AX, 256 → AX= (0000 0001 0000 0000)\(_2\)
  – → AH= (0000 0001)\(_2\) AL = (0000 0000)\(_2\)
  – ADDB AH, AL → AH= (0000 0001)\(_2\)
  – → AX= (0000 0001 0000 0000)\(_2\) = \((256)_{10}\)
Exercise

• B
  – What is the value in AX after executing the following instructions, where 256 is a decimal number?

    MOV AX, 256
    MULB AH, AL

• Solution
  – \((256)_{10} = (0000 0001 0000 0000)_2\)
  – MOV AX, 256 \(\Rightarrow\) AX = (0000 0001 0000 0000)_2
  – \(\Rightarrow\) AH = (0000 0001)_2 AL = (0000 0000)_2
  – MULB AH, AL \(\Rightarrow\) AH = (0000 0000)_2
  – \(\Rightarrow\) AX = (0000 0000 0000 0000)_2 = (0)_{10}
Exercise

• C
  – What is the value in AX after executing the following instructions, where 257 is a decimal number?

    MOV AX, 257
    MULB AH, AL

• Solution
  – \((257)_{10} = (0000 0001 0000 0001)_2\)
  – MOV AX, 257 \(\rightarrow\) AX= (0000 0001 0000 0001)\(_2\)
  – \(\rightarrow\) AH= (0000 0001)\(_2\) AL = (0000 0001)\(_2\)
  – MULB AH, AL \(\rightarrow\) AH= (0000 0001)\(_2\)
  – \(\rightarrow\) AX= (0000 0001 0000 0000 0001)\(_2\) = \((257)_{10}\)
Exercise

• D
  - What is the value in AX after executing the following instructions, where 258 is a decimal number?

    MOV  AX, 258
    MULB AH, AL

• Solution
  - \((258)_10 = (0000 0001 0000 0010)_2\)
  - \(\text{MOV } AX, 258 \Rightarrow AX = (0000 0001 0000 0010)_2\)
  - \(\Rightarrow AH = (0000 0001)_2 \ AL = (0000 0010)_2\)
  - \(\text{MULB AH, AL } \Rightarrow AH = (0000 0010)_2\)
  - \(\Rightarrow AX = (0000 0010 0000 0010)_2 = (514)_{10}\)
Exercise

• **E**
  – What is the value in AX after executing the following instructions, where 514 is a decimal number?

    MOV AX, 514
    DIVB AH, AL

• **Solution**
  – \((514)_{10} = (0000 0010 0000 0010)_{2}\)
  – MOV AX, 514 \(\rightarrow\) AX= (0000 0010 0000 0010) \(_{2}\)
  – \(\rightarrow\) AH= (0000 0010) \(_{2}\) AL = (0000 0010) \(_{2}\)
  – DIVB AH, AL \(\rightarrow\) AH= (00000001) \(_{2}\)
  – \(\rightarrow\) AX= (0000 0001 0000 0010) \(_{2}\) = \((258)_{10}\)
Exercise

• F
  – What is the value in AX after executing the following instructions, where 513 is a decimal number?

  MOV AX, 513
  DIVB AH, AL

• Solution
  – \((513)_{10} = (0000 0010 0000 0001)_2\)
  – MOV AX, 513 \(\rightarrow\) AX= \((0000 0010 0000 0001)_2\)
  – \(\rightarrow\) AH= \((0000 0010)_2\) AL = \((0000 0001)_2\)
  – DIVB AH, AL \(\rightarrow\) AH= \((00000010)_2\)
  – \(\rightarrow\) AX= \((0000 0010 0000 0001)_2 = (513)_{10}\)
Exercise

• G
  – What is the value in AX after executing the following instructions, where 1026 is a decimal number?

MOV AX, 1026
DIVB AH, AL

• Solution
  – \((1026)_{10} = (0000 0100 0000 0010)_2\)
  – MOV AX, 1026 \(\rightarrow\) AX= \((0000 0100 0000 0010)_2\)
  – \(\rightarrow\) AH= \((0000 0100)_2\) AL = \((0000 0010)_2\)
  – DIVB AH, AL \(\rightarrow\) AH= \((00000010)_2\)
  – \(\rightarrow\) AX= \((0000 0010 0000 0010)_2 = (514)_{10}\)
Pointer Registers (1)

- **Pointer or Index Registers**
  - Stack pointer, Base pointer, Source pointer, Destination pointer
- **Stack Pointer (SP)**
  - The stack is a segment of memory that holds certain context information about the running program. Usually, when a procedure is called, part of the stack is reserved for holding the procedure’s local variables, the address to return to when the procedure has finished, and other control information.
  - The portion of the stack relating to a procedure is called its stack frame. When a called procedure calls another procedure, an additional stack frame is allocated, usually just below the current one. Additional calls allocate additional stack frames below the current ones.
  - Stacks almost always grow downward, from high addresses to low addresses. But the lowest numerical address occupied on the stack is always called the top of the stack.
  - **PUSH** puts a 16-bit word on top of the stack. This instruction first decrements SP by 2, then stores its operand at the address SP is now pointing to. **POP** removes a 16-bit word from the top of the stack by fetching the value on top of the stack and then incrementing SP by 2.
  - The **SP** register points to the top of the stack and is modified by **PUSH**, **POP**, and **CALL** instructions, being decremented by **PUSH**, incremented by **POP**, and decremented by **CALL**.
Pointer Registers (2)

• **Base Pointer (BP)**
  – It usually contains an address in the stack. Whereas SP always points to the top of the stack, BP can point to any location within the stack.
  – In practice, a common use for BP is to point to the beginning of the current procedure’s stack frame, in order to make it easy to find the procedure’s local variables.
  – Thus, BP often points to the bottom of the current stack frame (the stack frame word with the highest numerical value) and SP points to the top (the stack frame word with the lowest numerical value).
  – The current stack frame is thus delimited by BP and SP.

• **Source Index (SI) and Destination Index (DI)**
  – SI, the source index, and DI, the destination index.
  – These registers are often used in combination with BP to address data in the stack, or with BX to compute the addresses of data memory locations.
Instruction Pointer and Segment Register

• Instruction Pointer (PC)
  – The instruction pointer has the Intel’s name: program counter (PC).
  – This register is not addressed directly by the instructions, but contains an address in the program code segment of the memory.
  – The instruction cycle starts by fetching the instruction pointed to by PC. This register is then incremented before the rest of the instruction is executed.
  – So, it points to the first instruction beyond the current one.

• Segment Register
  – The stack, the data and the instruction codes all reside in main memory, but usually in different parts of it. The segment registers govern these different parts of the memory, which are called segments.
  – There are four registers in the segment register group.
  – CS for the code segment register; DS for the data segment register,
  – SS for the stack segment register; ES for the extra segment register.
  – Most of the time, their values are not changed.
  – In practice, the data segment and stack segment use the same piece of memory, with the data being at the bottom of the segment and the stack being at the top.
Flag Register

• Flag Register
  – The flag register (condition code register) is a set of single-bit registers.
  – Some of the bits are set by arithmetic instructions and relate to the codes:
    Z - result is zero
    S - result is negative (sign bit)
    V - result generated an overflow
    C - result generated a carry
    A - Auxillary carry (out of bit 3)
    P - parity of the result
  – Other bits in this register control operation of certain aspects of the processor.
    – The I bit enables interrupts.
    – The T bit enables tracing mode, which is used for debugging.
    – Finally, the D bit controls the direction of the string operations.
  – Not all 16 bits of this flag register are used;
  – The unused ones are hardwired to zero.
Print

_EXIT = 1
_PRINTF = 127

.SECT .TEXT
start:
MOV AX, 2
MOV BX, 514
MUL BX

.output:
MOV CX, 0
PUSH CX
MOVB CL, BL
PUSH CL
MOVB CL, BH
PUSH CL
MOVB CL, AL
PUSH CL
MOVB CL, AH
PUSH CL
PUSH pfmf3
PUSHPRINTF
ADD SP, 4
POP CX
MOV CX, AH
POP CX
MOV CX, AL
ADD SP, CL

.exit:
PUSH 0
PUSH(EXIT
SYS

.SECT .DATA
pfmf3:
.ASCIIZ "%d, %d, %d, %d\n"

.SECT .BSS