Daniel Benjamin HW #2 01/22/2021

1. Name all 32-bit general purpose registers. What is the general function of each of the registers? Which of these registers cannot be addressed in parts?

32-bit registers:

① EAX (Extended AX):
- serves as an accumulator where arithmetic
- used for I/O port access, arithmetic, interrupt calls
- scratch pad register

② ECX (Extended CX):
- serves as a loop counter, also known as an iterator
- counts number of iterations inside a loop such

③ ESP (Extended SP):
- serves as a stack pointer, where locals, proc, and return addresses are stored
- holds top address of the stack

④ ESI (Extended Sarce Index):
- used for string and memory array copying

⑤ EDI (Extended Destination Index):
- used for string, memory array copying and setting and for far pointer addressing with ES register

⑥ EBX - (base register):
- used as a base pointer for memory access
- gets some interrupt return values

⑦ EBP (extended base pointer register):
- holds the base address for a stack

⑧ EDX (extended data register):
- used in input/output and most arithmetic instructions
- can be used along with EAX register for multiply and divide operations involving larger values

The following 32-bit registers cannot be addressed into parts:
- ESI
- EDI
- EBP
- ESP

These index and base general-purpose 32-bit registers including ESI, EDI, EBP, and ESP, cannot be further partitioned into 8-bit parts.

There is no way for the following index and base pointers to directly refer to their upper parts. For Example, the SI part of an ESI register can be referred but the upper part of the ESI register cannot be directly referred to since the upper part does not have a name. The same logic applies to the EDI, EBP, and ESP registers

2. What do the sign flag, zero flag, auxillary carry flag, and parity flag indicate when they are zero? What do the overflow flag and carry flag indicate when they are zero? How do OF and CF differ?

The carry flag (CF):
- carry is generated when performing n bit operations and the result is more than n-bits
- carry flag is also called the borrow flag
- If CF produces a 0, no carry out or borrow into MSB bit
- If CF produces a 1 carry out from MSB bit on addition or borrow into MSB bit on subtraction
- deals with unsigned arithmetic out of range

The sign flag (SF):
- indicates the sign for an signed number
- If SF = 0, MSB is zero meaning number is positive
- If SF = 1, MSB is one meaning number is negative

The zero FLAG (ZF):
- After any arithmetical or logical operation if the result is 0, the zero flag becomes set (1), otherwise it becomes reset (0)
- Z = 1 (zero result)
- Z = 0 (non-zero result)

The auxillary carry flag (AC):
- flag is used in BCD (binary coded decimal) number system
- If any arithmetic or logical operation results in a carry from bit 3 to bit 4, the flag becomes set(1), otherwise it becomes reset(0)
- only flag register not accessible by the programmer
- AC = 1 (carry out from bit 3 to bit 4)
- AC = 0 (otherwise)

Parity Flag (PF):
- If the result of any arithmetic or logical operation has even parity, an even number of 1 bits, the parity flag register becomes set(1), otherwise it becomes reset(0)
- P = 1 - accumulator has even number of 1 bits
- P = 0 - accumulator has odd number of 1 bits
- used for error detection

The overflow flag (OF):
- signed arithmetic out of range
- OF = 1, the result of a signed operation is too large to fit in the number of bits available to represent
- OF = 0, if the two most significants bits of a signed number both generate or do not generate carry, overflow flag reset

2. How OF and CF differ?
The main difference between of and CF is that OF deals with the signed number system and cf deals with the unsigned number system for arithmetic that is out of range

3. What is Cache memory and its benefits?

Cache memory is high-speed expensive static RAM located both inside and outside the CPU. When data to be read is already in cache memory, this is called a cache hit. When data to be read is not in cache memory, this is called a cache miss.

Cache memory is costlier than main memory or disk memory but is more economical than CPU registers. Cache memory is an extremely fast memory type that acts as a buffer between RAM and the CPU. Cache Memory holds frequently requested data and instructions so that they are immediately available to the CPU when needed. Cache memory is used to reduce the average time to access data from the main memory. It is possible to have different levels of cache memory, where level 1 cache would be closer to the CPU than a level 2 or level 3 cache. Level 1 cache would be faster than level 2 or level 3 cache all located inside the CPU.

4. What do you understand about Real-address mode, protected mode, and the Multi-segment model? Discuss in detail

Real-Address Mode:
- In Real-Address mode, the memory is partitioned into code and data segments and their
- this part of the memory contains kernel data such as I/O addresses, interrupt routines, and etc.

- programs are not protected from being overwritten, meaning there are no checks to who can write, read, and execute.

- native to MS-DOS (started around 1984)
  - programmers used different addresses while overwriting code with data causing MS-DOS to crash a lot
  - Microsoft did not fix protection issues until Windows XP came out

- 1 MB RAM maximum addressable
- Application programs can access any area of memory
- supports single tasking (interrupts)
- 1 MB = 2^{20} bytes = 2^{20} addresses
- each byte has a unique address
- RAM stands for Random Access Memory
- segmented memory addressing: absolute (linear) address is a combination of a 16-bit segment value added to a 16-bit offset
- net address size is 20-bits (address bus size = 16 bits)
- the offset part of segmented memory is the displacement of the particular address from its base address

4. Protected Mode
- 4GB addressable RAM (range of addresses 00000000 to FFFFFFFF)
- Each program is assigned a memory partition, providing protection from other programs and users.
- Designed for multi-tasking
- Supported by Linux and MS-Windows
- Program structure:
  - code, data, and stack areas, each with their own memory partition within the memory allocated for each program
  - CS, DS, SS segment descriptors
  - Code segment, data segment, stack segment
  - code segment contains a list of machine instructions
- Each segment within a program has their own built-in protections, preventing different segments such as CS, DS, and SS to overwrite each others memory

Multi-segment model:
- Each program has a local descriptor table (LDT)
  - holds descriptor for each segment used by a program
- A row within a LDT is called a descriptor containing the base address, the limit or amount of memory being allocated, and the access rights which are a combination of read(r), write(w), and execute(x)
- access rights determine whether a programmer can read, write, or execute code.
- the base address of a particular segment is the starting address for a segment
- the limit of a segment is the number of bytes in hexadecimal

5. In a 32-bit computer, what is the maximum amount of memory that can be addressed in (a) protected mode and (b) real-addressed mode

(a) protected mode:

- The maximum amount of memory that can be addressed considering 32-bit computer is 4 GB, which equals

$$4\,GB = 2^{32} = 2^2 \cdot 2^{30} = 4 \cdot 2^{30} \text{ addresses/bytes}$$

↙ answer

- The range of addresses/bytes for a 32 bit computer in protected mode is $(0 .. 2^{32} - 1)$

(b) real-addressed mode

answer ↓

- The maximum amount of memory that can be addressed in real-address mode is 1 MB. If one wants to access memory above 1 MB, protected mode must be used.

6. Let us say that the MOV instruction takes 4 clock periods on your computer. Express the time taken by the instruction in nanoseconds. Also the computer is running at 1.5 GHz.

$$T = \frac{1}{f} \qquad f = 1.5\,GHz = 1.5 \times 10^9\,Hz$$

$$nano = 1 \times 10^{-9}$$

$$T = \frac{1}{1.5 \times 10^9} = \frac{1}{1.5} \times 10^{-9} = \frac{1}{1.5}\,ns$$

In other words, it takes $\frac{1}{1.5}$ ns for the computer to complete one clock period while executing the MOV instruction.

Finding the net time for a computer running at a frequency of 1.5 GHz while executing the MOV instruction over 4 clock periods.

net time (ns) = (# of clock periods) · (amount of time to complete one cycle (ns))

$$\text{net time (ns)} = (4) \cdot \left(\frac{1}{1.5}\,ns\right) = \frac{4}{1.5}\,ns \approx \boxed{2.6667\,ns}$$

↑ answer