COSC2406

Assembly Language Programming

Assignment 2

Jeel Tikiwala, 239659420, 10th March 2024

Question 1. Short questions - type answers. (20 marks)

a) Name all five steps in the instruction execution cycle, briefly describe each and indicate the three basic required steps by placing [BASIC] at the end of the description. [4 marks]

a. FETCH
    i. The CPU fetches the instruction from the memory address called the instruction queue that is pointed to by the program counter (PC). It then increments the instruction pointer [BASIC].

b. DECODE
    i. The memory is decoded to understand what actions are required. The CPU decodes the instructions by looking at its binary bit pattern, which possibly contains operands which tells the CPU what operations to perform and with what data.

c. FETCH THE OPERANDS
    i. The CPU fetches the operand(s) needed for the execution if they are not already present in the instruction. This step may involve fetching data from registers and memory which require address calculations.

d. EXECUTE
    i. The CPU executes the instruction using any operands it fetched. This could be an arithmetic operation, a memory read or write, a jump, etc. ii.
    ii. It also updates the status flags like “Zero”, “Carry”, and “Overflow”. [BASIC].

e. WRITEBACK
    i. This execution is to register file or memory. This could be writing a value to a register or storing data back into memory. [BASIC].

b) The x86 processors have 4 modes of operation, three of which are primary and one sub-mode. Name and briefly describe each mode. [4 marks]

a. REAL ADDRESS MODE
    i. It is a simplest mode of operation and the state an x86-based PC is in when it is powered on.
    ii. It has the programming environment of an early intel processor with a few added features. (eg. Ability to switch into other modes.)

iii. This mode is useful if a program requires direct access to system memory, I/O and hardware devices as there is no concept of memory protection, multitasking, or code privilege levels.
iv. The processor acts like the original 8086 CPU, on which the x86 architecture is based.

b. PROTECTED MODE
   i. The native state of the processor where all instructions and features are available to be used (Full capacity of the processor).
   ii. Protected mode supports multitasking, virtual memory, and provides several protection mechanisms to control the execution of the code, including the use of privilege levels for tasks and the segmentation mechanism to isolate and protect memory areas.
   iii. Programs are given segments, and the processor prevents programs from referencing memory outside their assigned segments.

c. SYSTEM MANAGEMENT MODE
   i. SMM is a special-purpose operating mode providing handling system-wide functions like power management and system security.
   ii. The code for SMM is typically stored in a dedicated area of memory that is not accessible to the normal operating system environment.

d. VIRTUAL-8086 MODE (Sub Mode)
   i. This is a sub mode of Protected Mode.
   ii. In this mode, software that was written for the 8086 CPU can run in a virtual environment on a more modern CPU, with some limitations.

c)Name all eight 32-bit general purpose registers. Identify the special purpose of each register where one exists. [8 marks]

EAX (Extended accumulator register) -Used for arithmetic operations, I/O operations, string manipulations, and return values from function.	**EBP** (Extended Base/Frame pointer) -Used by high-level languages to reference function parameters and local variables on the stack
EBX(Extended Base Register**)** -Used to store base address of a data segment in segmented memory architecture.	**ESP** (Extended stack pointer) - Addresses data on the stack, and is rarely used for arithmetic or data transfer - Points to the top of the stack and is automatically updated by push and pop instructions, call, and return operations.
ECX(Extended Count Register**)**	**ESI** (Extended Source Index)

-Used by the CPU for loop counting	-Used by high-speed memory transfer instructions -Used for string and array operations but pointing to the source in stream operations.
EDX(Extended Data Register**)** - Used in arithmetic operations and I/O operations; in conjunction with EAX, it can be used for multiply/divide operations and for functions that return a large (more than 32-bit) value.	**EDI** (Extended destination index) -Used by high-speed memory transfer instructions - Similar to ESI, it is used for string and array operations, but it points to the destination in stream operations.

d) Name at least four CPU status flags discussed in class and briefly describe their purpose. [4 marks]

In X86 architecture, the CPU flags are parts of FLAGS register.

The **Carry flag** is set when the result of an unsigned arithmetic operation is too large to fit into the destination.

The **Overflow flag** is set when the result of a signed arithmetic operation is either too large or too small to fit into the destination.

1. Four CPU status flags besides the carry and overflow flag are:
   a. Sign flag
      i. The sign flag reflects the sign of the result of an arithmetic operation.
      ii. Set when the result of an arithmetic or logical operation generates a negative result.
   b. Zero flag
      i. This flag is set if the result of an operation is zero.
      ii. Set when the result of an arithmetic or logical operation generates a result of zero.
      iii. It is commonly used in conditional branch operations to check for equality or to see if an operation, like a subtraction, has resulted in a zero value.
   c. Auxiliary carry flag
      i. Set when an arithmetic operation causes a carry from bit 3 to bit 4 in an 8-bit operand.
      ii. It indicates if an arithmetic operation has resulted in a carry out of the most significant bit (for addition) or a borrow (for subtraction).

iii. This flag is often used for unsigned arithmetic operations and for multiple precision arithmetic.

d. Parity flag
   i. Set if the least-significant byte in the result contains an even number of 1 bits. (used for error checking when data may be altered or corrupted).
   ii. Conversely, it is cleared if the number of set bits is odd.

Question 2: SHOW ALL YOUR WORK (either in the same document or as a separate PDF scan). NO SUPPORT = 50% PENALTY. All your calculation work must be provided with the assignment.

a) **(10 marks)** Convert the real number stored in IEEE Single precision format to its base 10 value.
1 10000001 10010100000000000000000b

b) **(10 marks)** The IEEE-Double Precision format is similar to the IEEE-Single Precision format except that
the biased exponent is 11 bits instead of 8, the bias value is 1023 instead of 127, and the fractional component is 52 bits instead of 23. Using this information, store the number 422.32 into IEEE Double Precision format.

Jeel Tikiwala, 239659420

COSCC 2406 - Assembly language.

(a)

1 10000001 1001 0100 0000 000 0 0000 006

Sign bit = 1 (The number is -ve)

Exponent = 10000011

= 129 in decimal

: 129 - 127 = 2

Fraction = 1001 0100 0000 0000 0000 006

Formula = $(-1)^{sign} \times 1.mantissa \times 2^{exp-127}$

Mantissa = 1.1001 0100 0000 0000 0000 00

Binary to decimal

$1.100101_{(2)} = 1 \times 2^0 + 1 \times 2^{-1} + 0 \times 2^{-2} + 0 \times 2^{-3} + 1 \times 2^{-4} + 0 \times 2^{-5} + 1 \times 2^{-6}$

$= 1 + 0.5 + 0.0625 + 0.015625$

$\boxed{= 1.578125 \text{ (approx)}}$

$= (-1)^{sign} \times 1.mantissa \times 2^{exp}$

$= (-1)^1 \times 1.578125 \times 2^2$

$= -1 \times 1.578125 \times 4$

$= -1 \times 1.578125 \times 4$

$= -1.578125 \times 4$

$\boxed{= -6.3125}$

1578125
× 4
6312500

(b) (1) Convert to Binary

422.32

422 = 110100110 (in binary).

.32 x 2 = 0.64 → 0	0.48 x 2 = 0.96 → 0
.64 x 2 = 1.28 → 1	0.96 x 2 = 1.92 → 1
.28 x 2 = 0.56 → 0	0.92 x 2 = 1.84 → 1
.56 x 2 = 1.12 → 1	0.84 x 2 = 1.68 → 1
.12 x 2 = 0.24 → 0	0.68 x 2 = 1.36 → 1
.24 x 2 = 0.48 → 0	0.36 x 2 = 0.72 → 0
	0.72 x 2 = 1.44 → 1
$\boxed{.32 \approx 0.01010\ldots}$	0.44 x 2 = 0.88 → 0
	0.88 x 2 = 1.76 → 1
~~(2) Normalise~~:	0.76 x 2 = 1.52 → 1
	0.52 x 2 = 1.04 → 1
	0.04 x 2 = 0.08 → 0
	0.08 x 2 = 0.16 → 0
	0.16 x 2 = 0.32 → 0

$(0.32)_{10} = (0.01010001111010111000)_2$

Integral value is

$(422)_{10} = (0110100110)_2$

$(422.32)_{10} = (0110100110.01010001111010111$
$000)_2$

$(422.32)_{10} = (1.10100110 01 01000 1111 010111$
$000)_2 \times 2^8$

Mantissa is :

1010011 001 01000 1111 01 0111000

0000 0000 0000 0000 0000 0000 00

The no is positive, sign bit = 0

Exponent = bias + power of 2

= 1023 + 8

$\boxed{= 1031}$

$(1031)_{10} = (10000000111)_2$

Sign	Exponent	Mantissa
0	10000000111	1010 0110 01 01 000 1111010111000 0000 0000 0000 0000 0000 0000 00

0100 0000 0111 1010 0110 0101 0001 1110

1011 1000 0000 0000 0000 0000 0000 0000

$\boxed{0X\ 407A651EB8000000}$