**Assignment #2 Answer Sheet Name: Matthew Ferlaino**

**ENTER ALL YOUR FINAL ANSWERS ON THIS ANSWER SHEET FOR QUESTIONS 10 to 17 AND ATTACH YOUR SUPPORTING WORK USING EXTRA SHEETS AS A PDF**

**Question 1**

**5 Steps in the Execution Cycle:**

1. Fetch the Instruction – fetches the instruction from the instruction queue. It then will increment the instruction pointer.
2. Decodes the Instruction – decodes the instruction by looking at the binary bit pattern of the instruction.
3. Fetch the Operands(Optional) – if there are operands then the CPU will fetch them from memory and registers.
4. Executes – the CPU will execute the instruction and if needed use operand values it may have fetched. Will also update the status flag.
5. Stores the Result (Optional) – if the output operand was part of the instruction the CPU will store the result of the execution in the operand

**Question 2**

When a processor switches from one task to another we must preserve the first task’s program counter, task variables and CPU registers.

**Question 3**

1. *Protected Mode* – the native state of the processor, all instructions and features will be available. Programs are given segments in memory which allows the CPU to prevent programs from referencing memory outside of their assigned segments.
2. *Virtual-8086 Mode* – in this mode the processor can directly execute real-address mode software in a safe environment. What this means is that if a program crashes or tries to write data into the system memory area it cannot effect other programs running at the same time.
3. *Real-Address Mode* – will implement a programming environment of an early Intel processor with extra features like the ability to switch into other modes. Can be helpful if your program needs to have direct access to memory or hardware devices.
4. *System Management Mode* – this mode will provide the operating system with a mechanism for implementing functions such as power management and system security. The functions are usually implemented by the computer manufacturers who will customize the processor for a certain type of system setup because there can be different kinds of set ups.

**Question 4**

When running in 64-bit programming mode, two of the four modes of operation are not supported. These two modes would be real-address mode and virtual 8086 mode.

**Question 5**

The 64-bit x86-64 processors use a 48-bit physical address space which means that the CPU can support 2 to the power of 64 bits.

**Question 6**

**32-bit General Purpose Registers:**

1. EAX(Extended Accumulator Register) – used when multiplication and division happen.
2. EBX(Extended Base Register) – this register can be used in index addressing.
3. ECX(Extended Count Register) – the CPU will automatically use this as the loop counter
4. EDX(Extended Data Register) – usually used in combination with the EAX register for multiplication and division.
5. EBP(Extended Base Pointer Register) – registered is used in helping reference the parameter variables passed to a subroutine.
6. ESP(Extended Stack Pointer Register) – this register is used to address data on the stack
7. ESI(Extended Stack Index Register) – register is used by high-speed memory transfer instructions.
8. EDI(Extended Destination Register) - register is used by high-speed memory transfer instructions.

**Question 7**

When the result of an unsigned arithmetic operation is done and is too large to fit into the destination the “carry flag” will be set (will equal 1).

**Question 8**

When a signed arithmetic operation is either too large or too small to fit into the destination the “overflow flag” will be set (will equal 1).

**Question 9**

**4 Other Status Flags:**

1. Sign Flag(SF) –when the result of either an arithmetic or logical operation generates a negative result the sign flag is set
2. Zero Flag(ZF) – when the result of either an arithmetic or logical operation generates a result of zero the zero flag will be set
3. Auxiliary Carry Flag (AC) – will be set when an arithmetic operation causes a carry from bit 3 to bit 4 in an 8-bit operand
4. Parity Flag (PF) – this flag is set if the least-significant byte in the result contains an even number of 1 bits. This flag is usually empty, can be used for error checking when there is a possibility that data might be corrupt or altered

**Question 10 – Clock Cycle time in nanoseconds**

|  |  |  |
| --- | --- | --- |
| **a) .0000000004405** | **b) .000000000215** | **c) .000000000294** |

**Question 11 – TRUTH TABLE**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **A** | **B** | **C** | **AC** | **~B** | **AC + ~B** | **A · B** | **~C** | **A · B + ~C** |  |  | **(AC + ~B) Ꚛ (A · B + ~C)** |
| **0** | **0** | **0** | **0** | **1** | **1** | **0** | **1** | **1** |  |  | **0** |
| **0** | **0** | **1** | **0** | **1** | **1** | **0** | **0** | **0** |  |  | **1** |
| **0** | **1** | **0** | **0** | **0** | **0** | **0** | **1** | **1** |  |  | **1** |
| **0** | **1** | **1** | **0** | **0** | **0** | **0** | **0** | **0** |  |  | **0** |
| **1** | **0** | **0** | **0** | **1** | **1** | **0** | **1** | **1** |  |  | **0** |
| **1** | **0** | **1** | **1** | **1** | **1** | **0** | **0** | **0** |  |  | **1** |
| **1** | **1** | **0** | **0** | **0** | **0** | **1** | **1** | **1** |  |  | **1** |
| **1** | **1** | **1** | **1** | **0** | **1** | **1** | **0** | **1** |  |  | **0** |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |

**Question 12 – Parity Flag Value (true if it is set, and false if it is not)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#1 TRUE** | **#2 TRUE** | **#3 TRUE** | **#4 TRUE** | **#5 FALSE** | **#6 TRUE** |

**Question 13 – Segment:Offset addresses (Hint: answers will be 5 digit HEX numbers)**

|  |  |  |
| --- | --- | --- |
| **#1 FCDE4h** | **#2 700DBh** | **#3 A993Ah** |

**Question 14 - Equivalent SEG:OFF addresses (The format should be AAAA:AAAA)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A)** | **00000Fh** | **49800h** | **564C0h** | **553B0h** |
| **B)** | **0000Ah** | **6482Bh** | **725F0h** | **73710h** |

**Question 15 – Convert the Base 10 REAL to IEEE-Single Precision**

|  |  |  |
| --- | --- | --- |
| **Steps:** | **First Conversion** | **Second Conversion** |
| 1. **Sign bit** | **0** | **0** |
| 1. **Integer part in binary** | **00111100b** | **00000000b** |
| 1. **Fraction part in binary** | **10110b** | **01001111001b** |
| 1. **Normalized value in binary** | **0.011110010110 x 27** | **0.000000001001111001 x 27** |
| 1. **Biased exponent in binary** | **10000110b** | **10000110b** |
| 1. **IEEE format**   **(32 bits)** | **0 10000110 0011 1100 1011 0011 0011 001** | **0 10000110 0000 0000 0100 1111 0011 001** |

**Question 16 – Convert from IEEE-Single Precision to Base 10 REAL**

|  |  |  |
| --- | --- | --- |
| **Steps:** | **First Conversion** | **Second Conversion** |
| 1. **Sign** | **1** | **0** |
| 1. **Biased exponent** | **132d** | **126d** |
| 1. **Unbiased exponent** | **5d** | **-1d** |
| 1. **Normalized value in binary** | **1.011100100000 x 25** | **1.100000000000 x 2-1** |
| 1. **Unnormalized value** | **46.25d** | **0.305d** |
| 1. **Whole number in base 10** | **46d** | **0d** |
| 1. **Fraction in base 10** | **185/4d** | **3/8d** |
| 1. **Real number in base 10 (don’t forget the sign)** | **-46.25d** | **+0.305d** |

**Question 17 – Convert the Base 10 REAL to IEEE-Double Precision**

|  |  |
| --- | --- |
| **Steps:** |  |
| 1. **Sign bit** | **1** |
| 1. **Integer part in binary** | **10011010b** |
| 1. **Fraction part in binary** | **0110b** |
| 1. **Normalized value in binary** | **1.100110100110 x 28** |
| 1. **Biased exponent in binary** | **00010000111b** |
| 1. **IEEE format** 2. **(64 bits)** | **1 00010000111 1100 1101 0011 0011 0011 0011 0011 0011 0011 0011 0011 0011 0011** |

**ALL OF YOUR FINAL ANSWERS FOR QUESTIONS 10-17 MUST BE ON THIS FORM.**

**ATTACH YOUR SUPPORTING WORK – NO SUPPORT WILL RESULT IN A 50% PENALTY.**

**Support may be hand written and then scanned to a PDF document or typed.**