|  |  |  |  |
| --- | --- | --- | --- |
|  | **Ho Chi Minh City University of Technology**  **Department of Electrical and Electronics Engineering** | | |
| **FINAL EXAMINATION**  Grading: 40% | | | **Computer System Engineering**  Course ID: 407406 |
| **Date: 2 Jul, 2018** | | | **Duration:** 90 minutes |
| **Student name:**  **Student ID:** | | | **Examiner’s name & signature:** |
| **Score:** | | Students are allowed to use *one A4 page with two sides* for reference.  Books and other documents are not allowed to use. | |
| **This examination consists of 4 pages** | |

**Problem 1:** (10pts) Answer the following questions

1. Max clock of 8086 CPU is 5-10MHz x 16-20MHz 25-30MHz
2. Segment registers are SI, DI BP, SP CS, DS x
3. In an 8086 program, the segment:offset address is 35AF:12D1. Find the five-digit address:

36DC1

1. CPU 80386 which uses 32-bit physical address can manage :

a) 1 MB of memory

b) 1 GB of memory

c) 4 GB of memory x

1. Assume that we have the memory content as below.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Address | 0x8 | 0x9 | 0xA | 0xB |
| Content | 62 | 41 | FE | 23 |

What are the 32-bit data when we read a double-word at the address 0x8 with Little Endian mode? 23FE4162

**Problem 2:** (20pts) Answer the value of registers after the instruction is executed.

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Before** | **Instruction** | **After** |
| 1 | AX: F2 8C | mov AH, 128 | **EAX: 80 8C** |
| 2 | ECX: 00 12 23 5E  Value: DWORD | mov value, ecx | **ECX: 00 12 23 5E**  **value: 00 12 23 5E** |
| 3 | BX: CA 4E  CX: FF FF | add BX CX | **BX: CA 4D**  **CX: FF FF**  **SF:0 ZF:0 CF: 1 OF: 0** |
| 4 | AX: 7F FF | inc AX | **AX: 80 00**  **SF:1 ZF:0 OF:0** |
| 5 | EAX: FF FF FF F6  Doubleword at Double  FF FF FF D1 | imul eax, Double | **EAX: 00 00 01 D6**  **CF: 0 OF: 0** |

**Problem 3:** (10pts) Write 80x86 assembly language code for the following C procedure:

|  |  |
| --- | --- |
| **C procedure** | **ASM procedure**  *Assume that S is stored in EAX, num is store in EBX, i is stored in ECX* |
| int my\_func(int num) {  in i = 0;  sum = 0;  for (i=0; i<num; i++)  {  if(i<=10) sum = sum + 2\*i;  else  sum = sum + i;  }  return sum;  } | **my\_func:**  **mov eax, 0; // S = 0**  **mov ecx, 0; // i = 0**  **Loop1:**  **cmp ecx, ebx; // compare i and num**  **jnl exit\_loop;**  **cmp ecx, 10;**  **jl caculate1;**  **add eax, ecx; // sum = sum + i**  **inc ecx;**  **jmp Loop1;**  **caculate1:**  **imul ecx, 2;**  **add eax, ecx; // sum = sum + i\*2**  **inc ecx;**  **jmp Loop1;**  **exit\_loop:**  **ret;** |

**Problem 4:** (10pts) Write 80x86 assembly language code for the following C function. Assume that:

* a is stored in register EAX
* b is stored in register EBX
* rval is stored in register EDX

|  |  |
| --- | --- |
| int logical(int a, int b)  {  int t1 = a^b;  int t2 = t1 + 5;  int mask = (1<<13) - 5;  int rval = t2 & mask;  return rval;  } | logical:  mov ecx, eax // ecx = a  xor ecx, ebx; // t1 = a^b  add ecx, 5 // t2 = t1+5  shr edx, 13; // mask = >> 17  mov edx, 1; // edx = 1  shl edx, 13; // edx = 1 << 13  sub edx, 5; // ebx = (1 << 13) - 5  and edx, ecx; // rval = t2 & mask  ret |

**Problem 5:**  (10pts) Describe processing steps for interrupt handling?

|  |
| --- |
| Processing steps for interrupt handling:  1. Save state  Disable interrupts for the duration of the ISR or allow it to be interrupted too?  Save program counter  Save flags  Save register values  2. Jump to interrupt service routine  Location obtained by interrupt vector  3. Process interrupt  4. Restore state  Load PC, flags, registers etc. |

**Problem 6:** (10pts) Draw a circuit to demonstrate the external interrupts of 8086 processor, using PIC (Programmable Interrupt Controller) 8259, for three devices: Ethernet, keyboard, and SCSI Disk

|  |
| --- |
|  |

**Problem 7:** (10pts) Write 80x86 assembly language code that initialize vector 48H to point to the ISR

“isr48”. Assume that the CPU operates in Real Mode. Offset address of isr48 is 2358h, and segment address of isr48 is 0200h.

|  |
| --- |
| push ax  push ds  mov ax,0  mov ds,ax  mov ax, 2358h  mov [0120h],ax  mov ax, 0200h  mov [0122h],ax  pop ds  pop ax |

**Problem 8:** (10pts) 3. Explain **page hit** and **page** **fault** in paging technique.

|  |
| --- |
| If the valid bit (present bit) is 1, then the virtual page is in RAM, and you can get the physical page from the PTE. This is called a **page hit**, and is basically the same as a cache hit.  If the valid bit is 0, the page is not in RAM, and the 20 bit physical page is meaningless. This means, we must get the disk page corresponding to the virtual page from disk and place it into a page in RAM. This is called a **page fault**. |

**Problem 9:** (10pts) Write C++ code to manage memory

|  |  |  |
| --- | --- | --- |
| **No.** | **Requirement** | **C / C++ code** |
| 1 | - Declare a pointer A of integer.  - Provide memory allocation for 200 integers at the pointer A.  - Free memory at the pointer A | int \*A;  A = new int[200];  delete[] A; |
| 2 | - Declare a pointer B of char  - Provide memory allocation for 30 chars at the pointer B.  - Resize the memory allocation of B to 50 chars | char \*B;  B = new char[30];  B = realloc(B, 50\*sizeof(char)); |

*--------------------------------------------------- The end ------------------------------------------------------*

*Electronics Department Lecturer*

*Ass.Prof. Hoang Trang Dr. Truong QuangVinh*