Team #:____ Name:_____
Absent: 8-22-06

Below is the description of the hardware features of a typical desktop PC:

- Intel® Pentium® 4 processor 2.8GHz with 800MHz frontside bus
- 512MB PC2100 DDR SDRAM for multitasking power, expandable to 4.0GB
- Cache Memory 512KB L2 cache
- DVD+RW/CD-RW drive; 48x maximum speed CD-ROM drive
- 250GB (7200 rpm) Ultra DMA hard drive
- Graphics card: 512MB NVIDIA GeForce 7600 GS, TV-Out and 2 DVI
- 6 high-speed USB 2.0 ports and 2 IEEE 1394 (FireWire) ports
- Integrated 10/100Base-T network interface; V.92 high-speed data/fax modem
- USB optical mouse and HP Media Center keyboard
- Windows XP Media Center Edition 2005 operating system preinstalled

1) What does the processor do?

2) What is stored in main memory (RAM)?

3) What is stored on the hard disk?

4) What is the purpose of cache memory?

5) What terms relate to interconnection of internal PC components?

6) What terms relate to interconnection of external PC components?

7) What is a KB, MB, GB, MHz, GHz?

Team #:____ Name:_____

Absent: 8-22-06

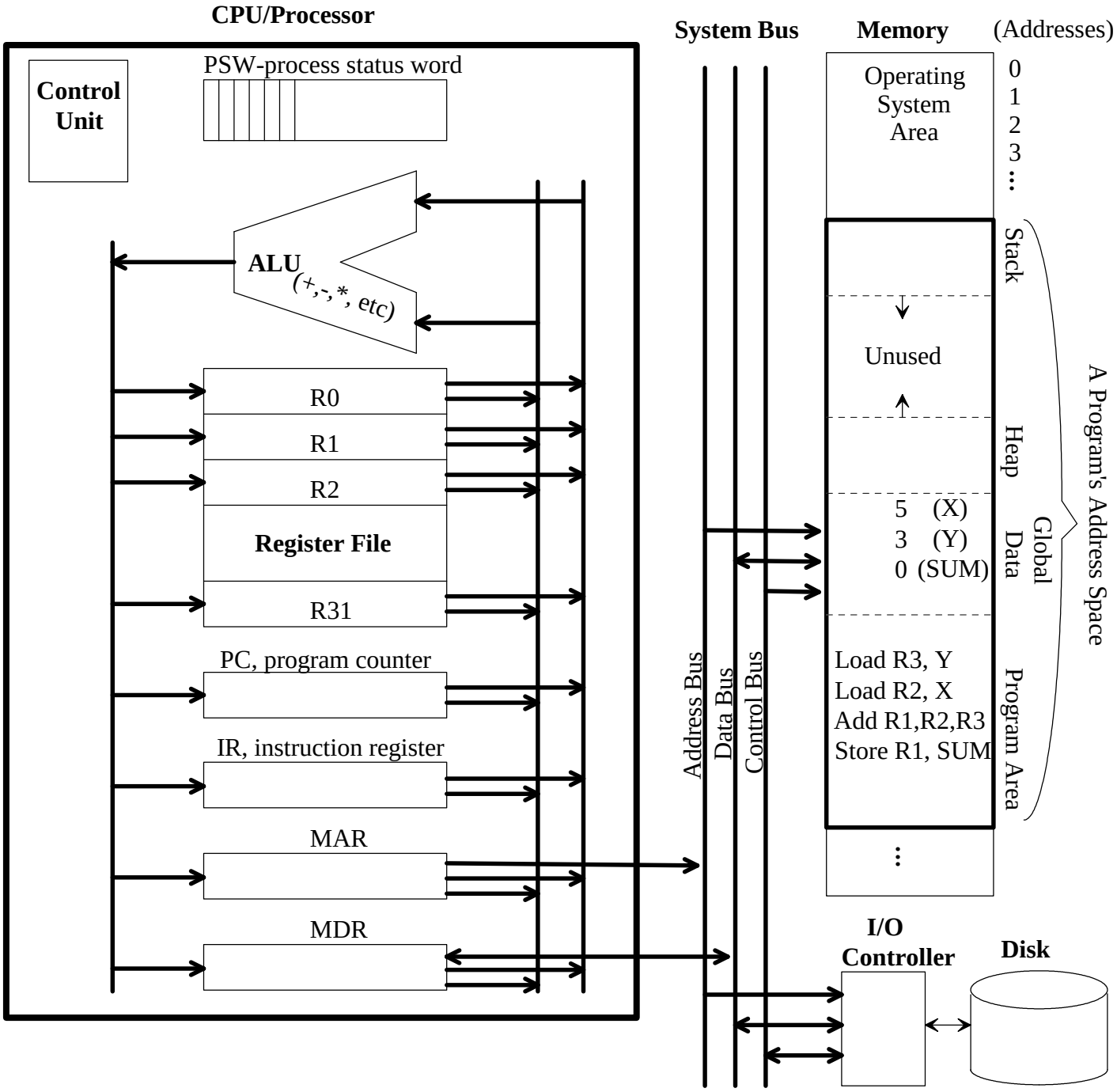
8) What is the role of a compiler?

9) What advantages do high-level languages (Ada, C, C++, Java, COBOL, etc.) have over assembly language?

10) Why do people program in assembly language (AL)?

Team #:____ Name:_____

Absent: 8-22-06



Instruction/Machine Cycle of stored-program computer - repeat all day

1. Fetch Instruction - read instruction pointed at by the program counter (PC) from memory into Instr. Reg. (IR)
2. Decode Instruction - figure out what kind of instruction was read
3. Fetch Operands - get operand values from the memory or registers
4. Execute Instruction - do some operation with the operands to get some result
5. Write Result - put the result into a register or in a memory location

(Note: Sometime during the above steps, the PC is updated to point to the next instruction.)

Team #:____ Name:_____

Absent: 8-22-06

Type of Instruction	**MIPS Assembly Language**	**Register Transfer Language Description**
Memory Access (Load and Store)	lw $4, Mem	$4← [Mem]
	sw $4, Mem	Mem←$4
	lw $4, 16($3)	$4← [Mem at address in $3 + 16]
	sw $4, Mem	[Mem at address in $3 + 16]← $4
Move	move $4, $2	$4← $2
	li $4, 100	$4← 100
Load Address	la $5, mem	$4← load address of mem
Arithmetic Instruction (reg. operands only)	add $4, $2, $3	$4← $2 + $3
	mul $10, $12, $8	$10← $12 * $8 (32-bit product)
	sub $4, $2, $3	$4← $2 - $3
Arithmetic with Immediates (last operand must be an integer)	addi $4, $2, 100	$4← $2 + 100
	mul $4, $2, 100	$4← $2 * 100 (32-bit product)
Conditional Branch	bgt $4, $2, LABEL (bge, blt, ble, beq, bne)	Branch to LABEL if $4 > $2
Unconditional Branch	j LABEL	Always Branch to LABEL

Fibonacci Sequence:	0	1	1	2	3	5	8	13	21
Position in Sequence:	0	1	2	3	4	5	6	7	8

A high-level language program to calculate the nth fibonacci number would be:

```
temp2 = 0
temp3 = 1

for i = 2 to n do
    temp4 = temp2 + temp3
    temp2 = temp3
    temp3 = temp4
end for

result = temp4
```

Trace of Program (time ⟶)

HLL variables								MIPS registers
temp2	0	1	1	2	3	5	8	$2
temp3	1	1	2	3	5	8	13	$3
temp4	1	2	3	5	8	13	21	$4
i	2	3	4	5	6	7	8	$1

A complete assembly language MIPS program to calculate the nth fibonacci number.

```
                    .data
n:                  .word  8          # variable in memory
result:             .word  0          # variable in memory

                          .text
                          .globl main
main:       li    $2, 0       # $2 holds temp2
            li    $3, 1       # $3 holds temp3
for_init:   li    $6, 2       # initialize i ($6) to 2
            lw    $5, n       # load "n" into $5
for_loop:   bgt   $6, $5, end_for
            add   $4, $2, $3  # $4 holds temp4
            move  $2, $3
            move  $3, $4
            addi  $6, $6, 1
            j     for_loop
end_for:
            sw    $4, result
            li    $v0, 10           # system code for exit
            syscall                 # call the operating system
```