ICS 233: Computer Architecture & Assembly Language
Fall Semester 2021 (211) – Section 01

Name: *ID#*

---

1. Fill in the blanks:
   a. The processor consists of two main units: _Datapath_ and _Control Unit_.
   b. Assemblers translates _Assembly instructions_ to _machine instructions/object files_.
   c. _Debugger_ allows you to trace the execution of your assembly code and set breakpoints.
   d. Pentium has a 32-bit address bus so its maximum physical address space it _2^{32} = 4G_ Bytes.
   e. Cache memory is faster than _main memory/disk_ but slower than _registers_.

2. Given the following data definitions, fill the data segment with **byte values** (as characters or hexadecimal numbers) using the **Big-Endian** byte ordering. Show also the mapping of the labels to their corresponding memory addresses in the symbol table, given that the starting address of **var1** is **0x10010000** (hexadecimal).

```
.data
var1: .ascii "ICS233\n"
var2: .half 0xabcd:2
var3: .space 5
.align 3
var4: .word 0x01234567
```

Label	Address
var1	0x10010000
var2	0x10010008
var3	0x1001000C
var4	0x10010018

Data Segment (Fill-in the bytes)

Byte	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0x10010000	'I'	'C'	'S'	'2'	'3'	'3'	'\n'		0xab	0xcd	0xab	0xcd				
0x10010010									0x01	0x23	0x45	0x67				
0x10010020																

ICS 233: Computer Architecture & Assembly Language
Fall Semester 2021 (211) – Section 02

Name: *ID#*

---

1. Fill in the blanks:
   a. The processor consists of two main units: _Datapath_ and _Control Unit_.
   b. Assembly language instructions has a one-to-one correspondence with _machine instructions_.
   c. _linker_ Combines object files created by the assembler with link libraries to generate a single executable file.
   d. MIPS has _32_ (how many?) general purpose registers. Each register is _32_-bit wide.
   e. Cache memory is bigger than _registers_ but smaller than _main memory/Disk_.

2. Given the following data definitions, fill the data segment with **byte values** (as characters or hexadecimal numbers) using the **little-Endian** byte ordering. Show also the mapping of the labels to their corresponding memory addresses in the symbol table, given that the starting address of **var1** is **0x10010000** (hexadecimal).

```
.data
var1: .byte 'I','C','S'
var2: .word 0x12345678:2
var3: .asciiz "Text"
.align 3
var4: .half 0xabcd
```

Label	Address
var1	0x10010000
var2	0x10010004
var3	0x1001000C
var4	0x10010018

Data Segment (Fill-in the bytes)

Byte	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0x10010000	‘I’	‘C’	‘S’		0x78	0x56	0x34	0x12	0x78	0x56	0x34	0x12	‘T’	‘e’	‘x’	‘t’
0x10010010	‘\0’								0xcd	0xab						
0x10010020																