Pre-Check

1.Please answer true/false to the following questions, and include an explanation:

(1).The compiler may output pseudoinstructions.

True. It is the job of the assembler to replace these pseudoinstructions.

(2).main job of the assembler is to generate optimized machine code.

False. That's the job of the compiler. The assembler is primarily responsible for replacing pseudoinstructions and resolving offsets.

(3).The object files produced by the assembler are only moved, not edited, by the linker.

False. The linker needs to relocate all absolute address references.

2 Assembling

Let's say that we have a C program that has a single function sum that computes the sum of an array. We've compiled it to the code , but we haven't assembled the code yet. Let's assume that the code for this program starts at address 0x00061C00

.data

	.data array: .word 1,2,3,4,5,6,7,8 .text
1 0x00061C00:	sum: la $t0, array
2 0x00061C08:	li $t1,8
3 0x00061C0C:	move $t2, $0
4 0x00061C10:	loop: blt $t1, $0, end
5 0x00061C14:	sll $t3, $t1, 2
6 0x00061C18:	add $t3, $t0, $t3
7 0x00061C1C:	lw $t3, 0($t3)
8 0x00061C20:	add $t2, $t2, $t3
9 0x00061C24:	addi $t1, $t1, -1
10 0x00061C28:	j loop
11 0x00061C2C:	end: move $a0, $t2
12 0x00061C30:	li $v0,1 # print_int
13 0x00061C34:	syscall

(1) Which lines contain pseudoinstructions that need to be converted to regular instructions? How?

1,2,3,4,11,12

la becomes the lui and ori instructions.

li becomes an addi instruction here (e.g. li t0, 4 → addi t0, x0, 4). move becomes an addi instruction (i.e. move rd, rs → addi rd, rs, 0).

(2) What is the program execution result ?

36

3 Addressing

We have several addressing modes to access memory (immediate not listed):

(a). Base displacement addressing adds an immediate to a register value to create a memory address (used for lw, lb, sw, sb).

(b). PC-relative addressing uses the PC and adds the immediate value of the

(1) What is the range of 32-bit instructions that can be reached from the current PC using a branch instruction?

A. Addresses between 0 and 64K -1

B. Addresses between 0 and 256K -1

C. Addresses up to about 32K before the branch to about 32K after

D. Addresses up to about 128K before the branch to about 128K after

D

(2)What is the maximum range of 32-bit instructions that can be reached from the current PC using a jump instruction?

A. Addresses between 0 and 64M -1

B. Addresses between 0 and 256M -1

C. Addresses up to about 32M before the branch to about 32M after

D. Addresses up to about 128M before the branch to about 128M after

E. Anywhere within a block of 64M addresses where the PC supplies the upper 6 bits

F. Anywhere within a block of 256M addresses where the PC supplies the upper 4 bits

F

(3)Given the following code (and instruction addresses), fill in the blank fields for the following instructions .

1 ox002cff00: loop: add $t1, $t2,$t0	| | | | | | 0x20|
2.0x002cff04: jal foo	|3 | |
3. 0x002cff08: bne $t1, $zero, foo	|5 | | | |
4. ...	
5. 0x002cff2c: foo: jr $ra	ra =

<table>
<tr><td>1 ox002cff00: loop: add $t1, $t2,$t0</td><td>| 0|10 |8 |9 |0 | 0x20|</td></tr>
<tr><td>2.0x002cff04:   jal   foo</td><td>|3 | 0x00B3FCB |</td></tr>
<tr><td>3. 0x002cff08:   bne $t1, $zero, foo</td><td>|5 |9 |0 | -3 |</td></tr>
<tr><td>4. ...</td><td></td></tr>
<tr><td>5. 0x002cff2c: foo: jr $ra</td><td>ra =0x002cff08</td></tr>
</table>

4. What is the assembly language statement corresponding to this machine instruction?
00af8020hex

```
Op        rs     rt      rd    shamt   funct
000000 00101 01111 10000 00000 100000
```

add $s0,$a1,$t7