[PDS1]
Decide the registers and their usage protocol:

The 32 general purpose registers have been assigned as follows:

Register Alias	Name	Usage
$r0	$zero	Constant zero
$r1	$at	Assembler temporary
$r2-$r3	$v0 - $v1	Function return values
$r4-$r7	$a0 - $a3	Argument passing
$r8-$r17	$t0- $t9	Temporaries (not preserved across calls)
$r18-$r25	$s0 - $s7	Saved registers (preserved across calls)
$r26	N/A	Reserved for os
$r27	N/A	Reserved for os
$r28	$gp	Global pointer
$r29	$sp	Stack pointer
$r30	$fp	Frame pointer
$r31	$ra	Return address

The division for floating point registers are given below. We will not be using special names for them but keep them as they are shown:

Name	Usage
$f0 - $f1	Function return values (float/double)
$f2 - $f3	Temporary or intermediate computations
$f4 - $f11	Function arguments (float/double)
$f12 - $f19	Temporary values (not preserved across calls)
$f20 - $f31	Saved values (preserved across calls)

[PDS2]
Decide upon the size for instruction and data memory:

Segment	Start address	Size	Description
Instruction Memory	0x00400000	16 KB	Holds program code (read-only during execution)
Data Memory	0x10010000	16 KB	Stores global/static variables and arrays
Stack	Grows down from 0x7FFFFFFC	Dynamic	Function calls, local vars, return addresses (grows down)

Memory details:

1. **Register Width**: 32-bit registers (standard MIPS).
2. **Word Addressing**: Aligned to 4-byte boundaries.
3. **Instruction Access**: Word-wise (i.e., PC += 4).

[PDS3]
Design the instruction layout for R-, I- and J-type instructions and their respective encoding methodologies:

1. **Arithmetic:**

Instruction	Type	Format	Opcode	Funct
add	R	opcode rs rt rd 00000 funct	000000	100000
sub	R	opcode rs rt rd 00000 funct	000000	100010
addu	R	opcode rs rt rd 00000 funct	000000	100001
subu	R	opcode rs rt rd 00000 funct	000000	100011
addi	I	opcode rs rt immediate	001000	—----
addiu	I	opcode rs rt immediate	001001	—----
madd	R	opcode rs rt 00000 00000 funct	011100	000000
maddu	R	opcode rs rt 00000 00000 funct	011100	000001
mul	R	opcode rs rt rd 00000 funct	011100	000010

2. **Logical**

Instruction	Type	Format	Opcode	Funct
and	R	opcode rs rt rd 00000 funct	000000	100100
or	R	opcode rs rt rd 00000 funct	000000	100101
andi	I	opcode rs rt immediate	001100	—-------
ori	I	opcode rs rt immediate	001101	—-------
not	R	opcode rs 00000 rd 00000 funct	011111	100111
xori	I	opcode rs rt immediate	001110	—-------
xor	R	opcode rs rt rd 00000 funct	000000	100110
sll	R	opcode 00000 rt rd shamt funct	000000	000000
srl	R	opcode 00000 rt rd shamt funct	000000	000010
sla	R	opcode 00000 rt rd shamt funct	011111	000001
sra	R	opcode 00000 rt rd shamt funct	000000	000011

3. **Data transfer and conditional branch:**

Instruction	Type	Format	Opcode	Funct
lw	I	opcode base rt offset	100011	—---------
sw	I	opcode base rt offset	101011	—---------
lui	I	opcode 00000 rt immediate	001111	—---------
beq	I	opcode rs rt offset	000100	—---------
bne	I	opcode rs rt offset	000101	—---------
bgt	I	opcode rs rt offset	011111	010001
bgte	I	opcode rs rt offset	011111	010010
ble	I	opcode rs rt offset	011111	010011

bleq	I	opcode rs rt offset	011111	010100
bleu	I	opcode rs rt offset	011111	010101
bgtu	I	opcode rs rt offset	011111	010110

4. **Unconditional Branch , Comparison,Floating point:**

Instruction	**Type**	**Format**	**Opcode**	**Funct**
j	J	opcode address	000010	—--------
jr	R	opcode rs 00000 00000 00000 funct	000000	001000
jal	J	opcode address	000011	—--------
slt	R	opcode rs rt rd 00000 funct	000000	101010
slti	I	opcode rs rt immediate	001010	—--------
seq	I	opcode rs rt immediate	011111	011000
mfc1	R	opcode rt 00000 rd 00000 funct	010001	000000
mtc1	R	opcode rt 00000 rd 00000 funct	010001	000001
add.s	R	opcode fmt ft fs fd funct	010001	000010
sub.s	R	opcode fmt ft fs fd funct	010001	000011
c.eq.s	R	opcode fmt ft fs fd funct	010001	110010
c.le.s	R	opcode fmt ft fs fd funct	010001	110111
c.lt.s	R	opcode fmt ft fs fd funct	010001	110101
c.ge.s	R	opcode fmt ft fs fd funct	010001	111000
c.gt.s	R	opcode fmt ft fs fd funct	010001	111001
mov.s	R	opcode fmt ft fs fd funct	010001	000110