1) 133 - 1
1. 0xYZ00021A
binar: ???? ?? 00 0000 0000 0000 0010 0001 1010
=> adr = 2 + 8 + 16 + 512 = 538 * 4 = 2152 ?
Pe formatul J adresele de memorie sunt pe 26 de biti, iar pe formatul I sunt doar pe 16 biti. ?
2. $sp: ($s1v2)($s2v2)($fpv2)$fp:(t)(v[i])($s3v1)($s2v1)($s1v1)($s0v1)($fpv1)$fp:(*v)(n)(x)(t)
Restaurarea cadrului de apel este necesara pentru a se reveni la valorile initiale dupa iesirea dintr-o procedura.
Daca $fp nu este restaurat in procedura putere el va indica catre val lui t si v[i], in loc de val adresei lui v, val lui n, x si t.
Daca nu se restaureaza $ra se pierde adresa la care trebuie sa se intoarca procedura, creandu-se astfel un ciclu infinit.
3. Da
Da ?

2) 133 - 2
1. 0x00A1E0TU
binar: 0000 0000 1010 0001 1110 0000 10?? ????
=> format R, rs = 5 = $a1, rt = 1 = $at, rd = 28 = $gp, shift amount = 2
2. O adresa de memorie se poate incarca cu lw pe stiva deoarece aceasta este un intreg pe 32 de biti.
lb $t0, ch
subu $sp, 4
sw $t0, 0($sp)
3. Programul afiseaza adresa la care este stocat caracterul din bb ?
In loc de lw $t0, bb in main punem lb $t0, bb.

3) 135 - 1
1. 0x000F81UT
binar: 0000 0000 0000 1111 1000 0001 00?? ????
=> format R, rs = 0 - val imediata, rt = 15 = $t9, rd = 16 = $s0, shift amount = 4
2. $reg reprezinta adresa de memorie, iar 0($reg) reprezinta val aflata la acea adresa de memorie.
Trei moduri de a parcurge sirurile de caractere:
- prin atribuirea lui $t0 val de la 0 la lung(sir) si accesarea elem de la sir($t0)
- prin atribuirea lui $t0 adresei sirului si accesarea elem prin 0($t0), 4($0) s.a.m.d
- prin atribuirea lui $t0 adresei sirului si incrementarea lui cu 4 cat timp elem de la adresa 0($t0) este dif de “\0”
3. a) F - nu se poate determina adancimea maxima a stivei ?
b) A - addu este o operatie cu val imediata, deci apartine clasei I ?
c) A - pe formatul intern I adr se reprez pe 16 biti

4) 135 - 2
1. nume $8, 24($11), op = 0x2B
binar: 1010 1101 0110 1000 0000 0000 0001 1000
In locul lui 24 poate fi pusa orice valoare x divizibila cu 4, pentru care exista un elem la adresa x($11)
Cea mai mare adresa care poate fi accesata de instr j este 2^26 * 4 = 2^28 ?
2. $sp:(0)($fpv6)$fp:(0)(1)($fpv5)$fp:(1)(2)($fpv4)$fp:(2)(3)($fpv3)$fp:(3)(4)($fpv2)$fp:(4)($s0v)($fpv1)$fp:(5)
adancime max: 18
Restaurarea cadrului de apel este importanta pentru recuperarea val pe care le avea reg $s, dar si reg $fp si eventual $ra.
3. Reprez stiva si retinerea raportarii la $fp cu - (sa stii cate valori ai adaugat pe stiva).

5) 141 - 1
1. 0xYZ090014
binar: ???? ??01 0000 1001 0000 0000 0001 0100
=> rs = 8 = $t0, rt = 9 = $t1 , add = 20 * 4 = 80 ?
2. $sp:($s1v2)($s0v2)($fpv2)$fp:(x)(v[i]+y)($s4v1)($s3v1)($s2v1)($s1v1)($s0v1)($fpv1)$fp:(*v)(n)(x)(y)(z)
Restaurarea cadrului de apel este necesara pentru a se reveni la valorile initiale dupa iesirea dintr-o procedura.
Daca $fp nu este restaurat in procedura exactXDivizori el va indica catre val lui x si v[i]+y, in loc de val adresei lui v, val lui n, x, y si z.
Daca nu se restaureaza $ra se pierde adresa la care trebuie sa se intoarca procedura, creandu-se astfel o bucla.
3. addu $sp, 8 are rolul de a scoate 2 elem de pe stiva.

6) 141 - 2
1. 0x010A80TU
binar: 0000 0001 0000 1010 1000 0000 00?? ????
  => rs = 8 = $t0, rt = 10 = $t2 , rd = 16 = $s0 , shift amount = 0
2. $sp:($s2v2)($s1v2)($s0v2)($fpv2)$fp:(a)(c)(v[i])($s4v1)($s3v1)($s2v1)($s1v1)($s0v1)($fpv1)$fp:(*v)(n)(a)(c)(t)
Restaurarea cadrului de apel este necesara pentru a se reveni la valorile initiale dupa iesirea dintr-o procedura.
Daca $fp nu este restaurat in procedura f el va indica catre val lui a, c si v[i], in loc de val adresei lui v, val lui n, a, c si t.
Daca nu se restaureaza $ra se pierde adresa la care trebuie sa se intoarca procedura, creandu-se astfel o bucla.
3. a) F - in reprez J adr se reprezinta pe 26 de biti iar 26 nu e multiplu de 4
b) A - 2^5 = 32
c) A - fiecare operatie are campul func diferit fata de restul
d) A - b e operatie de clasa I, iar j e operatie de clasa J

7) 131 - 1
1. rs = $t1 = 9 = 01001
rt = $t9 = 25 = 11001
adr: 100
binar: ???? ??01 0011 1001 0000 0000 0000 0100
0xXY390004, unde octetul Y are ultimii 2 biti 01
2. li $t1, 0 #contor
li $t2, 1 #putere
for:
beq $t1, 7, exit
mul $t2, $t2, $t0
addi $t1, $t1, 1
j for
exit:
move $a0, $t2
li $v0, 1
syscall
3. Registrii a nu se restaureaza la sfrasitul procedurii deoarece ei oricum sunt folositi doar cand vrem sa afisam sau sa citim un sir de caract, adica doar la apelurile de sistem. ?

8) 131 - 2
1. 0x030B00TU
binar: 0000 0011 0000 1011 0000 0000 00?? ????
  => rs = 24 = $s8, rt = 11 = $t3, rd = val imediata, shift amount = 0
2.
3. Etichetele main si et au acceasi adresa deoarece nu exista alte intrusctiuni intre ele.
Trebuie sa adaug o intructiune intre main si et.
et: 0x10100001

9) 132 - 1
1. rd = $t0 = 8 = 01000
rs = $s0 = 16 = 10000
rt = $s1 = 17 = 10001
clasa R => op: 0000 00
func: 10 0000
binar: 0000 0010 0001 0001 0100 0000 0010 0000
0x02114020
2. lw $a0, $sp
li $v0, 1
syscall
analog $fp
Afisam inainte si dupa apelarea procedurii valorile lui $ra, $s0 - $s7 si $fp.
3. (credit goes to MARA ca altfel imi da ea mie li $v0, 10 syscall la viata)
Nu merge pt orice reprez binara b1 si b2, cu b1 != b2.
Contra exemplu:

li $t1, 2 ; lli $t2, 2
add $t3, $t1, $t2 => $t3 = $t4
mul $t4, $t1, $t2

10) 132 - 2
1. 0xAE020013
binar: 1010 1110 0000 0010 0000 0000 0001 0011
=> rs = 16 = $s0, rt = 2 = $v0, adr = 19 * 4 = 76
2. $sp: ($s1v2)($s0v2)($fpv2)$fp:(x)(y)($s2v1)($s1v1)($s0v1)($fpv1)$fp:(*v)(*w)(n)
Daca $fp nu este restaurat in procedura media_aritmetica el va indica catre val lui x si y, in loc de val adresei lui v, w si val lui n.
Daca nu se restaureaza $ra se pierde adresa la care trebuie sa se intoarca procedura, creandu-se astfel un ciclu infinit.
3. Da, deoarece procesorul MIPS este unul pe 32 de biti, iar 8 bytes = 4*8 = 32 biti.

11) 142 - 1
1. rd = $8 = 01000
rs = $9 = 01001
rt = $17 = 10001
format R => op: 0000 00
shift amount = 00000
binar: 0000 0001 0011 0001 0100 0000 00?? ????
0x013140XY, unde octetul X are primi 2 biti 00
2. li $a0, 'A'
li $v0, 1
syscall
3. 2^16 - 1 = F4 - 2 (pt cunoscatori)

12) 142 - 2
1. clasa J
adr = 400/4 = 100 = 1100100
binar: ???? ??00 0000 0000 0000 0000 0110 0100
0xXY000064, unde octetul Y are ultimii 2 biti 00
2. $sp:($s2v2)($s1v2)($s0v2)($fpv2)$fp:(v[i])(m)(p)($s3v1)($s2v1)($s1v1)($s0v1)($fpv1)$fp:(*v)(n)(m)(p)
Daca $fp nu este restaurat in procedura exact_p_desc el va indica catre val lui v[i], m si p, in loc de val adresei lui v, val lui n, m si p.
Daca nu se restaureaza $ra se pierde adresa la care trebuie sa se intoarca procedura, creandu-se astfel un ciclu infinit.
3. 2 metode de parcurgere a tablourilor unidimensionale:
- prin atribuirea lui $t0 val de la 0 la lung(vector) si accesarea elem de la vector($t0) (folosit pt parcurgerile in for)
- prin atribuirea lui $t0 val de la n la 0 si lui $t1 adresei sirului si incrementarea lui cu 4, accesand elem de la adresa 0($t1), cat timp $t0 > 0 (folosit pt parcurgerea in proceduri repetitive)

13) 134 - 1
1. format I
rs = $a0 = 4 = 00100
rt = $t5 = 13 = 01101
adr = 200/4 = 50 = 110010
binar: ???? ??00 1000 1101 0000 0000 0011 0010
0xXY8D0032, unde octetul Y are ultimii 2 biti 00
2. move $a0, $sp
div $a0, $a0, 4
li $v0, 1
syscall
3.

14) 134 - 2
1. clasa R => op: 0000 00
rd = $v1 = 3 = 00011
rs = $s1 = 17 = 10001

rt = $t0 = 8 = 01000
binar: 0000 0010 0010 1000 0001 1??? ???? ????
0x02281XYZ, unde octetul X are primul bit 1
func: 10 0000 => add

2. move $t0, $fp
move $t1, $sp
sub $a0, $t0, $t1
div $a0, $a0, 4 ?
li $v0, 1
syscall
3.

15) 144 - 1
1. li $t1, 2 ; lli $t2, 2
add $t3, $t1, $t2 => $t3 = $t4
mul $t4, $t1, $t2
2. $sp:($s0v2)($fpv2)$fp:(v[i]+t)($s3v1)($s2v1)($s1v1)($s0v1)($fpv1)$fp:(*v)(n)(x)(t)
Daca $fp nu este restaurat in procedura prima_cifra_para el va indica catre val lui v[i] + t, in loc de val adresei lui v, val lui n, x si t.
Daca nu se restaureaza $ra se pierde adresa la care trebuie sa se intoarca procedura, creandu-se astfel un ciclu infinit.
3. et: 0x10100001
Da, punand mai multe intructiuni inaintea lor. ?

16) 144 - 2
1. 0x000DE0TU
binar: 0000 0000 0000 1101 1110 0000 10?? ????
  => rs = 0 = $zero, rt = $13 = $t5, rd = 28 = $gp, shift amount = 2
sllv ?
2. Da, punand mai multe instructiuni inaintea ei.
3. Instr b se foloseste pentru branch-uri (daca...atunci sari la et), in timp ce instr j este folosita pt a se executa un jump (sari la et). ?