BİLGİSAYAR MİMARİSİ

Yıliçi Sınavı - İKİNCİ Öğretim

1) Aşağıdaki tabloda bulunan pseudo-komutlar için gerçek MIPS makine dili komutları karşılıklarını bulunuz? **[20p]**
$at register, geçici değer atamaları için kullanılabilir…

Pseudo-instruction	Solution
`clear $t5`	
`li $t5, imm32`	
`beq $t5, imm32, Label`	
`bge $t5, $t3, Label`	

2) Aşağıda verilen MIPS yazılımının yaptığı fonksiyonun C dilindeki ifadesini yazınız? **[20p]**
$25 : 5 sabit değerini tutar
$4: x dizisinin başlangıç adresini tutar $20: y dizisinin başlangıç adresini tutar

```
X: lw $10, 0($4)
lw $11, 4($4)
add $12, $11, $10
sw $12, 0($20)
addi $20, $20, 4
addi $4, $4, 4
subi $25, $25, 1
bne $0, $25, X
```

3) Tek Çevrimli (Single Cycle) Veriyolu yapısı üzerinde JALR komutu için üretilmesi gerekli kontrol işaretleri tablo 1 ve tablo 2 de görülmektedir. Buna göre şekilde verilen donanım üzerinde eksik bağlantıları gösteriniz? **[20p]**

Return Adress = PC + 4,

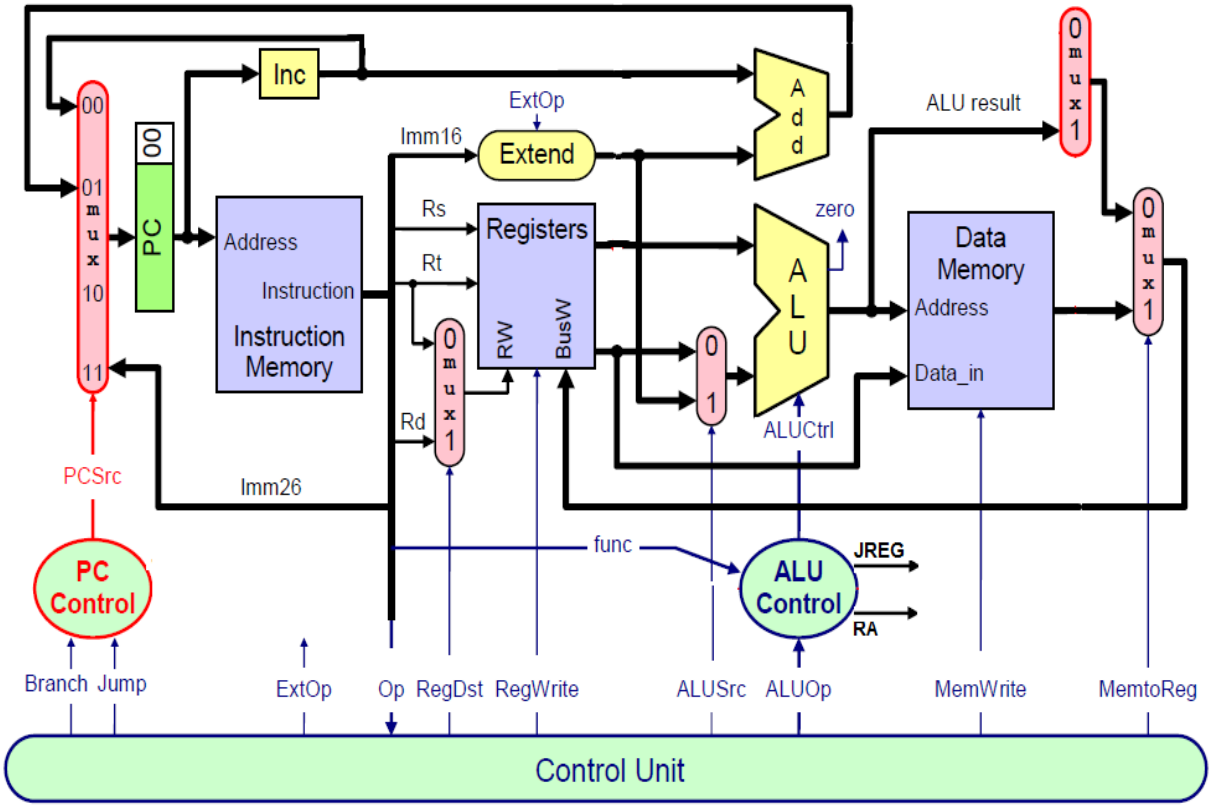
RA: Dönüş Adres Kontrol işareti
JREG : Jump Register (Rs) Kontrol İşareti

TABLO 1.. **JALR** Komutu için Kontrol İşaretleri

Instr.	RegDst	RegWrite	ALUSrc	ALUOp	MemWrite	MemtoReg	Branch	Jump
JALR	Rd = 1	1	Rt = 0	R-type	0	0	0	0

TABLO 2. **ALU Kontrol** İşaretleri

ALUOp	func	JReg	RA	ALUCtrl
R-type	JALR	1	0	X



4) Aşağıda makine kodları verilen programı MIPS komutları ile yazınız? [15p]

0x40 10001100101101110000000000100100

0x44 00000010111001001011000000100011

0x48 00011110110000001111111111110000

5) Register $s0 içinde bir n pozitif tamsayısı bulunduğunu düşünün. Fibonacci 1,1,2,3,5,8,13,… dizisi olarak bilinen diziyi hesaplayan programı MIPS komutları kullanarak yazınız? Toplam sonucu register $t2 da tutulacaktır.. [25p]

Category	Instr	Op Code	Example	Meaning
Arithmetic (R & I format)	add	0 and 32	add $s1, $s2, $s3	$s1 = $s2 + $s3
	subtract subtract unsigned	0 and 34 0 and 35	sub $s1, $s2, $s3 subu $s1, $s2, $s3	$s1 = $s2 - $s3 $s1 = $s2 - $s3
	add immediate	8	addi $s1, $s2, 6	$s1 = $s2 + 6
	or immediate	13	ori $s1, $s2, 6	$s1 = $s2 v 6
Data Transfer (I format)	load word	35	lw $s1, 24($s2)	$s1 = Memory($s2+24)
	store word	43	sw $s1, 24($s2)	Memory($s2+24) = $s1
	load byte	32	lb $s1, 25($s2)	$s1 = Memory($s2+25)
	store byte	40	sb $s1, 25($s2)	Memory($s2+25) = $s1
	load upper imm	15	lui $s1, 6	$s1 = 6 * 2^{16}
Cond. Branch (I & R format)	br on equal	4	beq $s1, $s2, L	if ($s1==$s2) go to L
	br on not equal	5	bne $s1, $s2, L	if ($s1 !=$s2) go to L
	set on less than	0 and 42	slt $s1, $s2, $s3	if ($s2<$s3) $s1=1 else $s1=0
	set on less than immediate	10	slti $s1, $s2, 6	if ($s2<6) $s1=1 else $s1=0
	brunch on greater than zero	7	bgtz $s, offset	If $s>0 PC= PC+4+4xsign_ext(imm16) else PC=PC+4
Uncond. Jump (J & R format)	jump	2	j 2500	go to 10000
	jump register	0 and 8	jr $t1	go to $t1
	jump and link	3	jal 2500	go to 10000; $ra=PC+4

MIPS KOMUT FORMATLARI

Instruction Formats

- All instructions are 32-bit wide, Three instruction formats:
- Register (R-Type)
  - Register-to-register instructions
  - Op: operation code specifies the format of the instruction

- Immediate (I-Type)
  - 16-bit immediate constant is part in the instruction

- Jump (J-Type)
  - Used by jump instructions

YANITLAR

BİLGİSAYAR MİMARİSİ

İKİNCİ ÖĞRETİM

ÇÖZÜMLER

1- a) 4P

clear $t5

~~addu $t1, $t2, $zero~~ 4P

subu $t5, $ti, $ti

Li $t5, imm 32

b) 5P

Lui $t5, upper 16 2P

ori $t5, $t5, lower 16 3P

c) 6P

beq $t5, imm 32, Label

Lui $at, upper 16 2P

ori $at, $at, Lower 16 2P

beq $t5, $at, Label 2P

d) 5P

bge $t5, $t3, Label

SLt $at, $t5, $t3 2P

beq $at, $zero, Label 3P

2)

```
X: Lw   $10, 0($4)      # $10 = x(0)
   Lw   $11, 4($4)      # $11 = x(1)
   add  $12, $11, $10   # $12 = x(0) + x(1)
   sw   $12, 0($20)     # y(0) = x(0) + x(1)
   addi $20, $20, 4     # $20 = $20 + 4
   addi $4, $4, 4       # $4 = $4 + 4
   subi $25, $25, 1     # $25 = $25 - 1
   bne  $0, $25, X      # if $25 = 0, goto X
```

20P

```
for (i=0 ; i<5 ; i++)
        y(i) = x(i) + x(i+1)
```

---

4)

adres				
0x40	100011	00101	10111	0000000000100100
	23 hex	5	17 (hex)	36 (dec.)
	opcode	↓	↓	↓
	↓	Rs	23 (dec)	Imm 16
	Lw		↓	
			Rt	

5P

Lw Rt, Imm 16 (Rs)

lw $23, 36($5) 5p

b) 0x44 | 000000 | 10111 | 00100 | 10110 | 00000 | 100011

00	23 dec	4 dec	22 dec.	0	35
↓	↓	↓	↓	Samt	↓
opcode	Rs	Rt	Rd	no shift	subu
R type					function code

5p

subu Rd, Rs, Rt

subu $22, $23, $4 5p

c) 0x48 | 000111 | 10110 | 00000 | 1111111111110000

opcode	22 dec	0	FFF0
7	↓	↓	↓
bgtz	Rs	Rt	Imm 16
	↓		000F (1's comp
			0010 (2's comp
			↓
			−16

5p

bgtz Rs, ~~offset~~

bgtz $22, −16 5p

5) Fibinoçci Dizisi

1, 1, 2, 3, 5, 8, 13 - - -

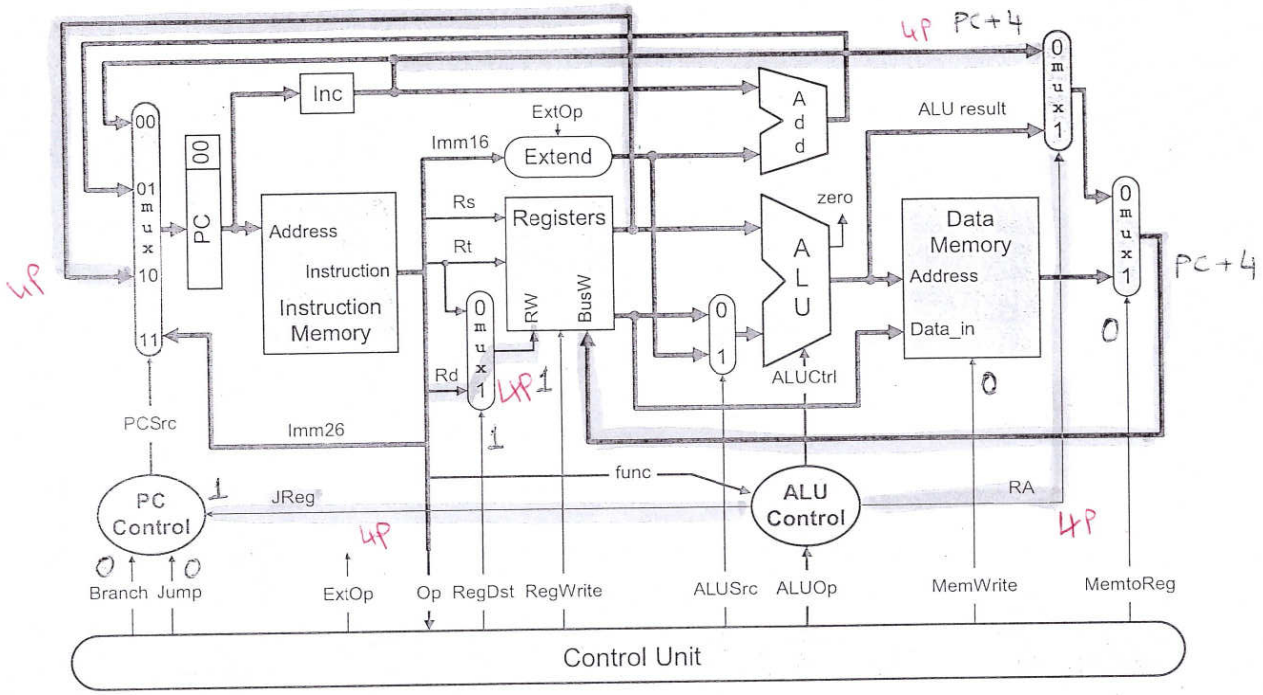
$s0 = n (dizinin sonu)

$t2 = sonuç toplam

```
li $s1, 1          # $s1 = 1
li $t1, 1          # $t1 = 1
li $t0, 0          # $t0 = 0
DON: add $t2, $s1, $t1     # $t2 = $t1 + $s1
move $t1, $s1      # $t1 = $s1
move $s1, $t2      # $s1 = $t2
add $t0, $t0, 1    # $t0 = $t0 + 1
blt $t0, $s0, DON  # $t0 < $s0 (=n) GOTO DON
```

2 P

3)



JALR Rs ⇒ PC ← ⟨RS⟩ jump adresi

Rd ← PC+4 dönüş adresi (ana program)