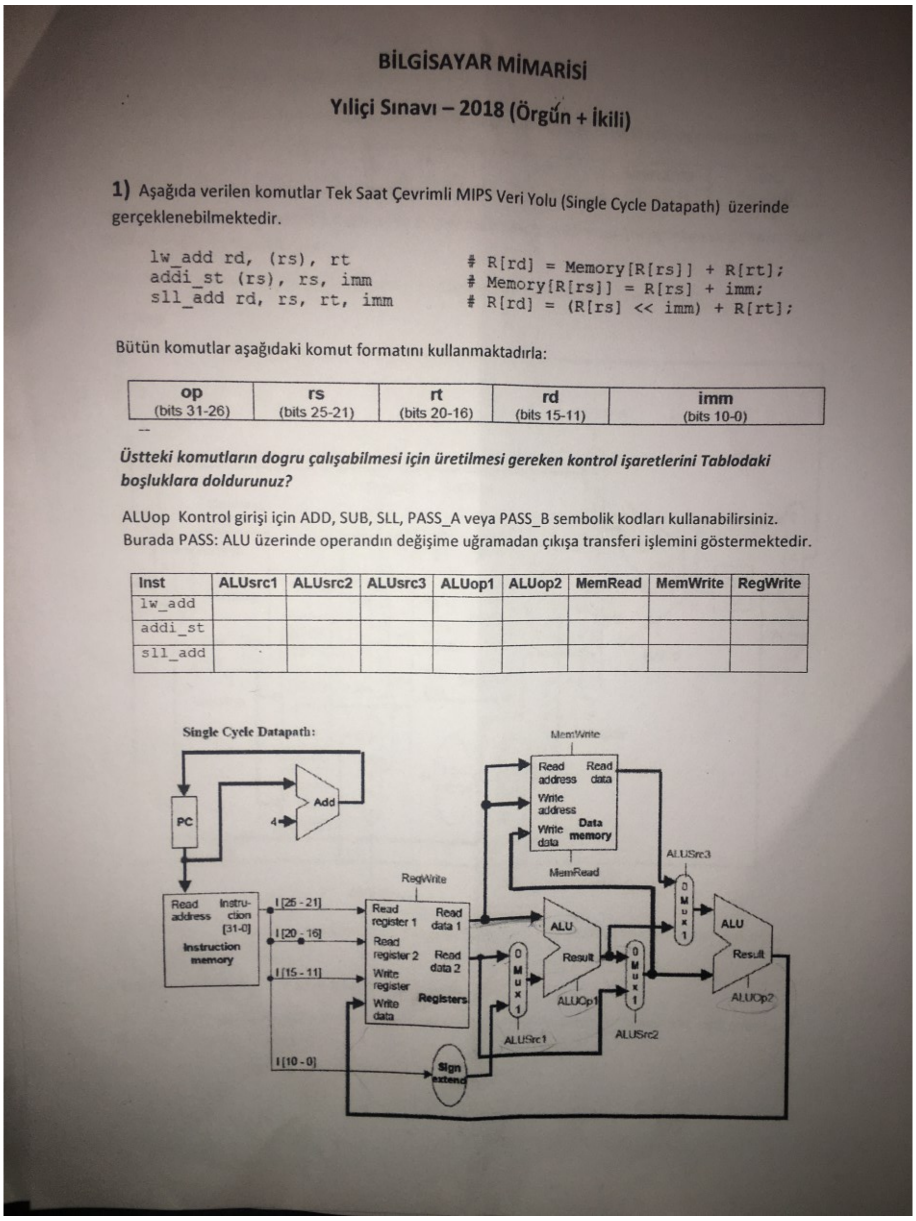
BİLGİSAYAR MİMARİSİ – 2018- ÖRGÜN + İKİLİ YILİÇİ SINAVI



BİLGİSAYAR MİMARİSİ

Yıliçi Sınavı – 2018 (Örgün + İkili)

1) Aşağıda verilen komutlar Tek Saat Çevrimli MIPS Veri Yolu (Single Cycle Datapath) üzerinde gerçeklenebilmektedir.

```
lw_add rd, (rs), rt            # R[rd] = Memory[R[rs]] + R[rt];
addi_st (rs), rs, imm          # Memory[R[rs]] = R[rs] + imm;
sll_add rd, rs, rt, imm        # R[rd] = (R[rs] << imm) + R[rt];
```

Bütün komutlar aşağıdaki komut formatını kullanmaktadırla:

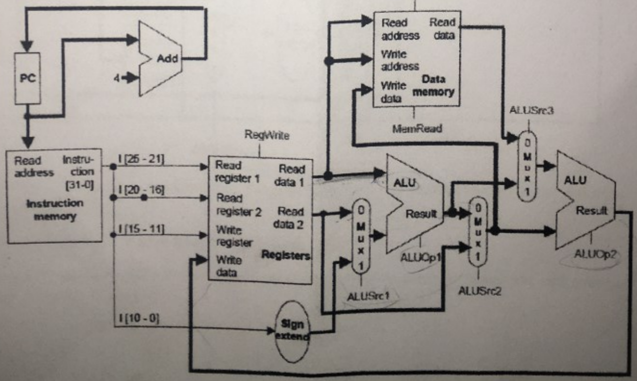
op (bits 31-26)	rs (bits 25-21)	rt (bits 20-16)	rd (bits 15-11)	imm (bits 10-0)

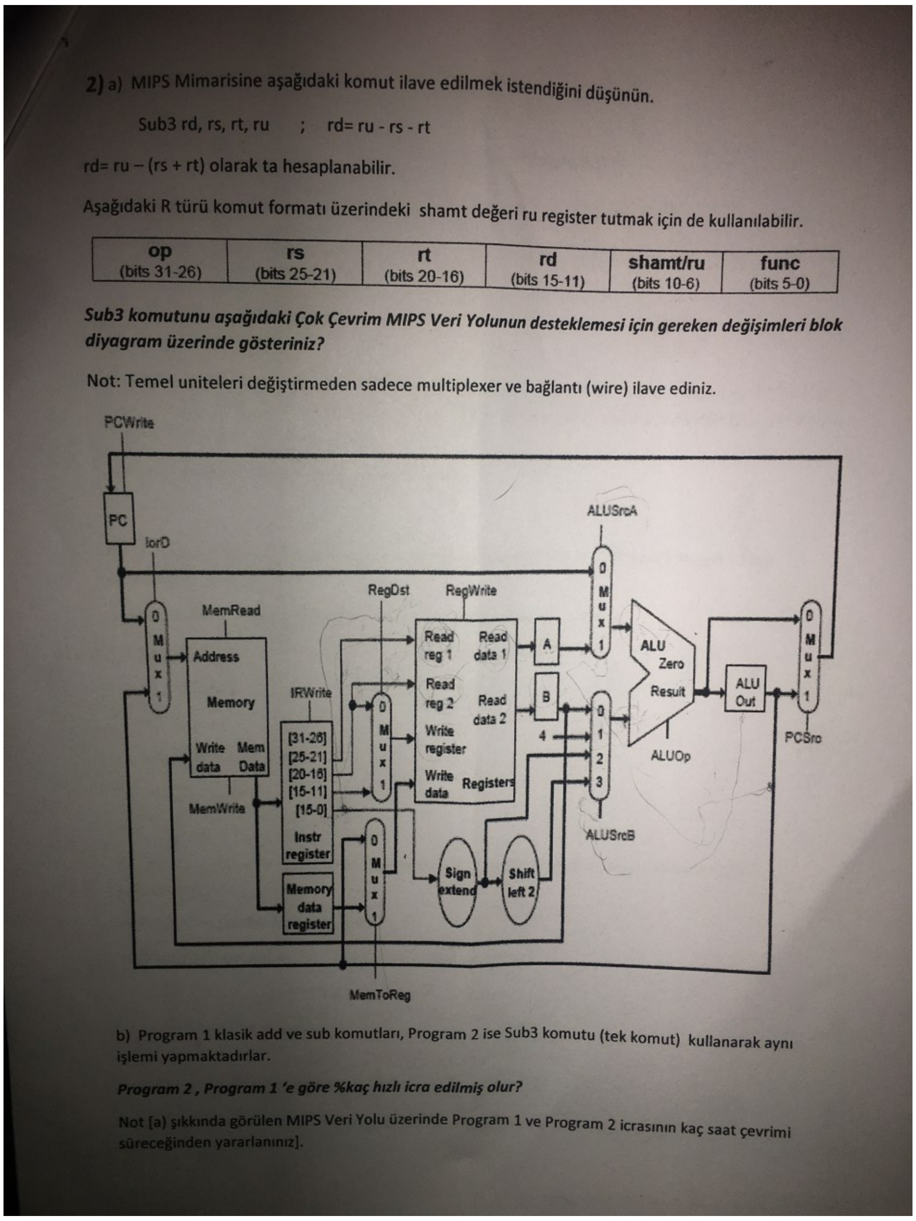
Üstteki komutların dogru çalışabilmesi için üretilmesi gereken kontrol işaretlerini Tablodaki boşluklara doldurunuz?

ALUop Kontrol girişi için ADD, SUB, SLL, PASS_A veya PASS_B sembolik kodları kullanabilirsiniz. Burada PASS: ALU üzerinde operandın değişime uğramadan çıkışa transferi işlemini göstermektedir.

Inst	ALUsrc1	ALUsrc2	ALUsrc3	ALUop1	ALUop2	MemRead	MemWrite	RegWrite
lw_add								
addi_st								
sll_add								

Single Cycle Datapath:





2) a) MIPS Mimarisine aşağıdaki komut ilave edilmek istendiğini düşünün.

Sub3 rd, rs, rt, ru ; rd= ru - rs - rt

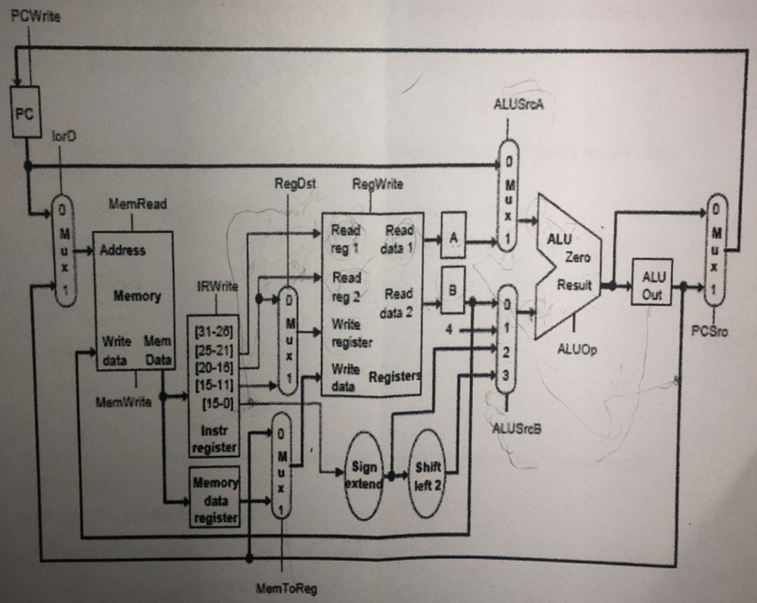
rd= ru – (rs + rt) olarak ta hesaplanabilir.

Aşağıdaki R türü komut formatı üzerindeki shamt değeri ru register tutmak için de kullanılabilir.

op	rs	rt	rd	shamt/ru	func
(bits 31-26)	(bits 25-21)	(bits 20-16)	(bits 15-11)	(bits 10-6)	(bits 5-0)

Sub3 komutunu aşağıdaki Çok Çevrim MIPS Veri Yolunun desteklemesi için gereken değişimleri blok diyagram üzerinde gösteriniz?

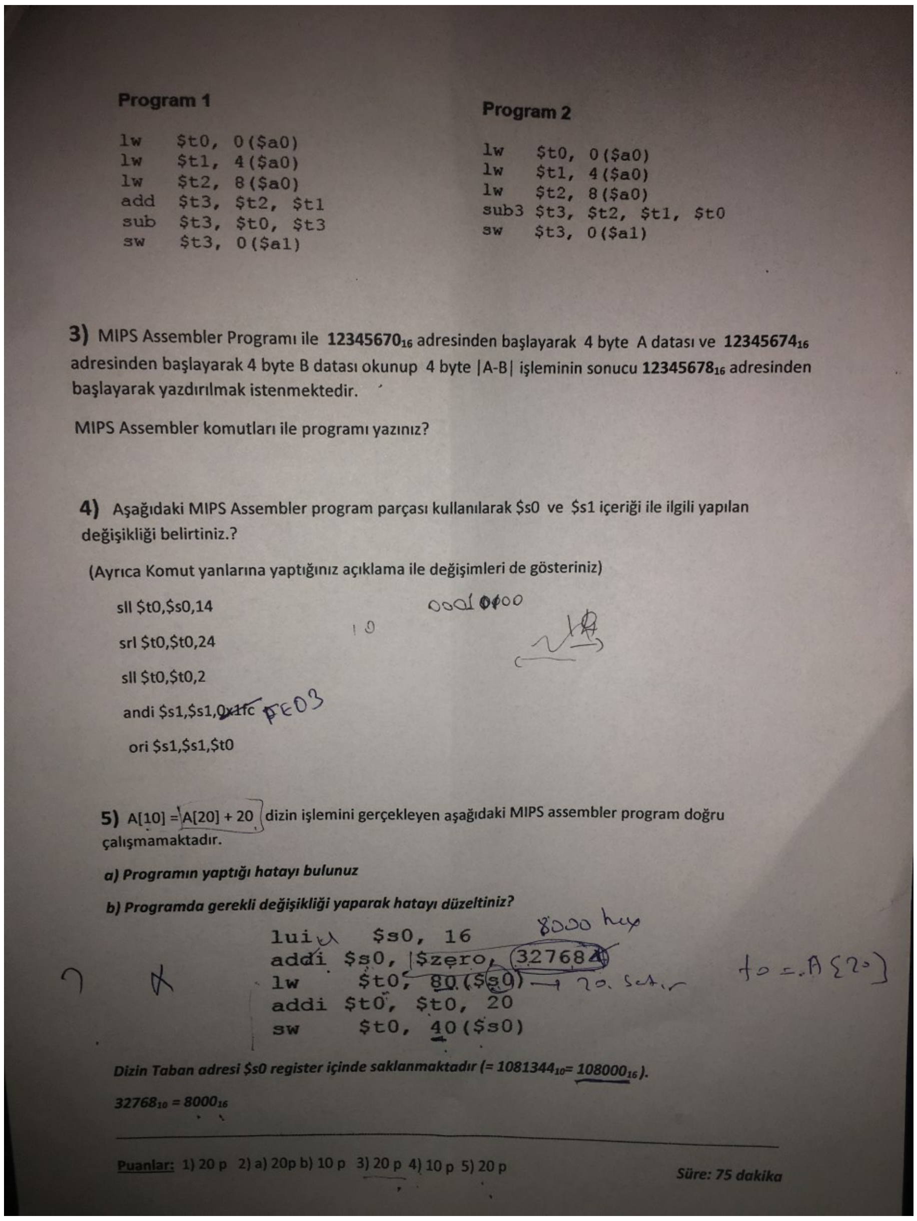
Not: Temel uniteleri değiştirmeden sadece multiplexer ve bağlantı (wire) ilave ediniz.



b) Program 1 klasik add ve sub komutları, Program 2 ise Sub3 komutu (tek komut) kullanarak aynı işlemi yapmaktadırlar.

Program 2 , Program 1 'e göre %kaç hızlı icra edilmiş olur?

Not [a) şıkkında görülen MIPS Veri Yolu üzerinde Program 1 ve Program 2 icrasının kaç saat çevrimi süreceğinden yararlanınız].



Program 1

```
lw   $t0, 0($a0)
lw   $t1, 4($a0)
lw   $t2, 8($a0)
add  $t3, $t2, $t1
sub  $t3, $t0, $t3
sw   $t3, 0($a1)
```

Program 2

```
lw   $t0, 0($a0)
lw   $t1, 4($a0)
lw   $t2, 8($a0)
sub3 $t3, $t2, $t1, $t0
sw   $t3, 0($a1)
```

3) MIPS Assembler Programı ile **12345670_{16}** adresinden başlayarak 4 byte A datası ve **12345674_{16}** adresinden başlayarak 4 byte B datası okunup 4 byte |A-B| işleminin sonucu **12345678_{16}** adresinden başlayarak yazdırılmak istenmektedir.

MIPS Assembler komutları ile programı yazınız?

4) Aşağıdaki MIPS Assembler program parçası kullanılarak $s0 ve $s1 içeriği ile ilgili yapılan değişikliği belirtiniz.?

(Ayrıca Komut yanlarına yaptığınız açıklama ile değişimleri de gösteriniz)

```
sll $t0,$s0,14          10          0001 0000
srl $t0,$t0,24
sll $t0,$t0,2
andi $s1,$s1,~~0x1fc~~ FE03
ori $s1,$s1,$t0
```

5) A[10] = A[20] + 20 dizin işlemini gerçekleyen aşağıdaki MIPS assembler program doğru çalışmamaktadır.

a) Programın yaptığı hatayı bulunuz

b) Programda gerekli değişikliği yaparak hatayı düzeltiniz?

```
                                8000 hex
lui   $s0, 16
addi  $s0, $zero, 32768
lw    $t0, 80($s0) → 20. satır        t0 = A[20]
addi  $t0, $t0, 20
sw    $t0, 40($s0)
```

Dizin Taban adresi $s0 register içinde saklanmaktadır (= 1081344_{10}= 108000_{16}).

$32768_{10} = 8000_{16}$

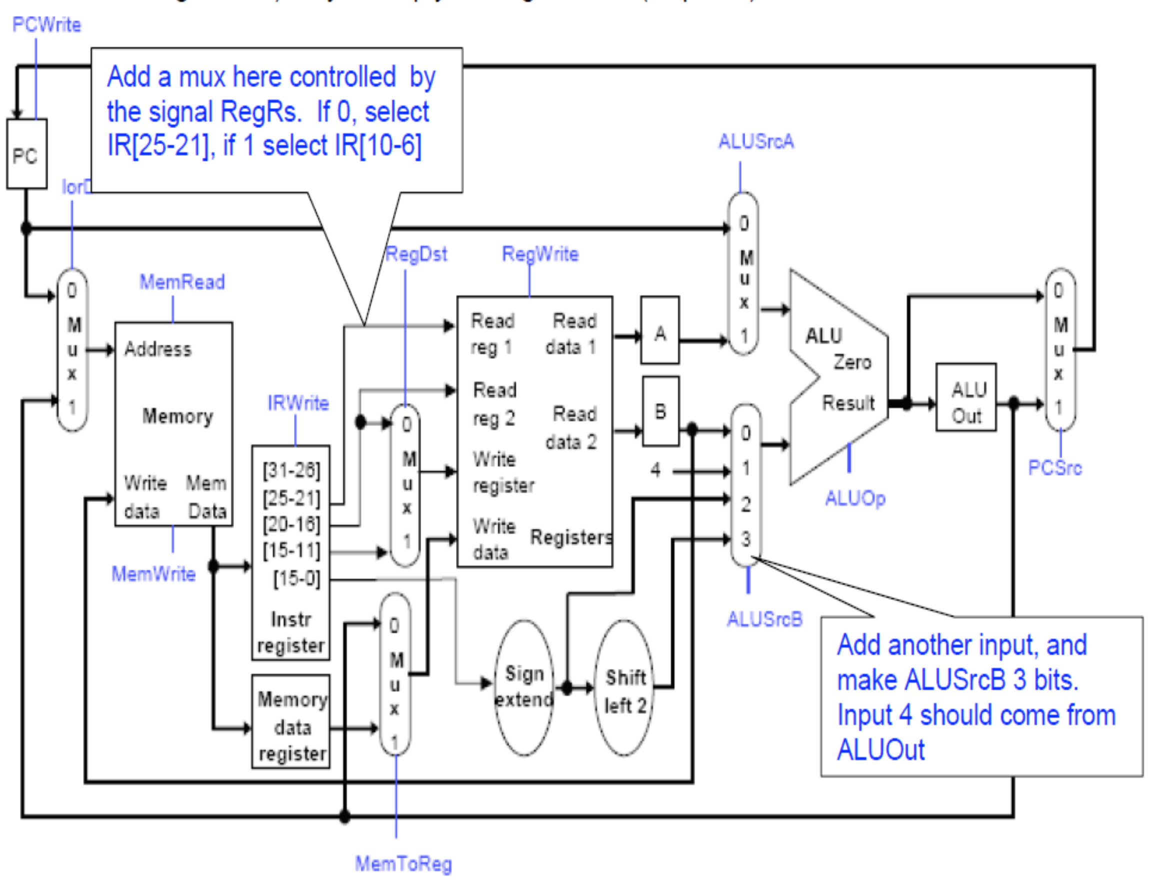
Puanlar: 1) 20 p 2) a) 20p b) 10 p 3) 20 p 4) 10 p 5) 20 p

Süre: 75 dakika

1-

Inst	ALUsrc1	ALUsrc2	ALUsrc3	ALUop1	ALUop2	MemRead	MemWrite	RegWrite
lw_add	X	1	0	X	ADD	1	0	1
addi_st	1	0	X	PASS_B	ADD	0	1	0
sll_add	1	1	1	SLL	ADD	0	0	1

2) a)



b)

Program 1	27 cycles
Program 2	24 cycles

Program 2 is faster by 3/27

)

3)

Program to calculate Absolute value of difference between 2 input numbers: |A - B| (demonstrates if)

Program reads A from 4 bytes of memory starting at address 12345670_{16}.

Program reads B from 4 bytes of memory starting at address 12345674_{16}.

Program writes |A-B| to 4 bytes of memory starting at address 12345678_{16}.

```
Assembler                    # Comment
lui   $10, 0x1234
ori   $10, $10, 0x5670      # put address of A into register $10
lw    $4,  0($10)           # read A from memory into register $4
lw    $5,  4($10)           # read B from memory into register $5 (A address+4)
sub   $12, $5, $4           # subtract A from B => B-A into register $12
bgez  $12,+1                # branch if B-A is positive to 'sw' instruction
sub   $12, $4, $5           # subtract B from A => A-B into register $12
sw    $12, 8($10)           # store register $12 value, |A-B|, into memory
```

4)

Write a sequence of no more than six MIPS instructions that extracts bits 17:11 of register $s0 and inserts them into bits 8:2 of register $s1, leaving all the remaining bits of $s1 unchanged. You may use $t registers as temporaries.

```
sll   $t0,$s0,14      # turn bits 17:11 of $s0 into bits 8:2 of $t0
srl   $t0,$t0,24
sll   $t0,$t0,2
# everything else in $t0 should be 0
andi  $s1,$s1,0xfe03  # zero out bits 8:2 in $s1
ori   $s1,$s1,$t0
```

5) a)

ANSWER

As a result of

```
    lui     $s0, 16
    addi $s0, $zero, 32768
```

instructions, the content of `$s0` is 1015808 (1048576 - 32768), since sign extension in

```
                  addi $s0, $zero, 32768
```

turned 32768 into - 32768

b)

```
lui  $s0, 16
ori  $s0, $zero, 32768
lw   $t0, 80($s0)
addi $t0, $t0, 20
sw   $t0, 40($s0)
```