BİLGİSAYAR MİMARİSİ

1) Aşağıdaki soruları MIPS Assembler programlama ile yanıtlayınız?

A) **abs** Rdest, Rsrc komutu Rdest ← ‖ $Rsource$ ‖ Rsrc register içeriğinin mutlak değeri

abs pseudo komutunu gerçek (real) makine komutları kullanarak gerçekleyiniz?

B) Aşağıdaki ifadeyi MIPS assemler dili karşılığını yazınız?
a, b, c, ve d sırasıyla $s0, $s1, $s2, ve $s3 register içinde tutulmuş olsunlar.
Bütün sayılar 32 bit işaretli sayılar olduğu kabul edilecektir... || or işlemi

if ((a > b) || (b > c)) {d = 1;}

2) Aşağıdaki MIPS Assembler programı bir dizi üzerinde işlem yaparak sonuçları $v0 ve $v1 registerler içinde tutmaktadır. Dizin 5000 words içermektedir (0 ile 4999 olarak indekslenen), ve dizin taban adresi $a0 içinde , ve kelime sayısı (5000) $a1 içinde tutulduğunu düşünün. .
1 Cümle halinde programın ne yaptığını tanımlayın, $v0 ve $v1 içinde tutulan değerlerin ne olduğunu belirtiniz?

```
        add $a1, $a1, $a1
        add $a1, $a1, $a1
        add $v0, $zero, $zero
        add $t0, $zero, $zero
outer:  add $t4, $a0, $t0
        lw $t4, 0($t4)
        add $t5, $zero, $zero
        add $t1, $zero, $zero
inner:  add $t3, $a0, $t1
        lw $t3, 0($t3)
        bne $t3, $t4, skip
        addi $t5, $t5, 1
skip:   addi $t1, $t1, 4
        bne $t1, $a1, inner
        slt $t2, $t5, $v0
        bne $t2, $zero, next
        add $v0, $t5, $zero
        add $v1, $t4, $zero
next:   addi $t0, $t0, 4
        bne $t0, $a1, outer
```

3) ALU çıkışında yapılan işlemin sonucunda taşma olup olmadığını test etmek amaçlı bir MIPS assembler program parçası yazınız.?

Not: Toplama işlemi yaptırarak Taşma olup olmadığı kontrol edilebilir..
Taşma varsa 'overflow' etiketine dallandırılacaktır.

4) MIPS mimarisinde komut çevrim adımları için işlem süreleri aşağıda verilmiştir.
IF = 7 ns, ID = 8 ns, EX = 15 ns, MA = 10 ns, WB = 8 ns
Pipeline registering 2 ns olsun…

a) Tek çevrimli MIPS datapath çevrim süresi (cycle time) nedir?
b) Çoklu çevrimli MIPS datapath çevrim süresi (cycle time) nedir?
c) Pipelined datapath çevrim süresi (cycle time) nedir?
d) Şıklarda hesapladığınız süreleri kullanarak ardarda 4 adet add komutunu (data bağımlılığı olmadığı kabulüyle) icra etmek için gereken süreleri i), ii) ve iii) için hesaplayınız?
   i) Tek Çevrimli MIPS datapath kullanarak,
   ii) Çok Çevrimli MIPS datapath kullanarak,
   iii) Pipelened MIPS datapath kullanarak

5) Şekilde görünen veri yolu tek saat çevrimlik işlemler için tasarlanmıştır. Register File R1'den R31'e kadar 31 adet saklayıcı içermektedir. PC = 20, R1= 5, R2 = 7 başlangıç değerleri verilmiştir. Diğer saklayıcıların içerikleri bilinmemektedir. Fetch işleminde getirilip işlenecek bir sonraki komut: "R2 ile gösterien bellek alanındaki içeriği R1'e yükle" dir.

LD R1, M[R2]

41	2	1	tanımsız

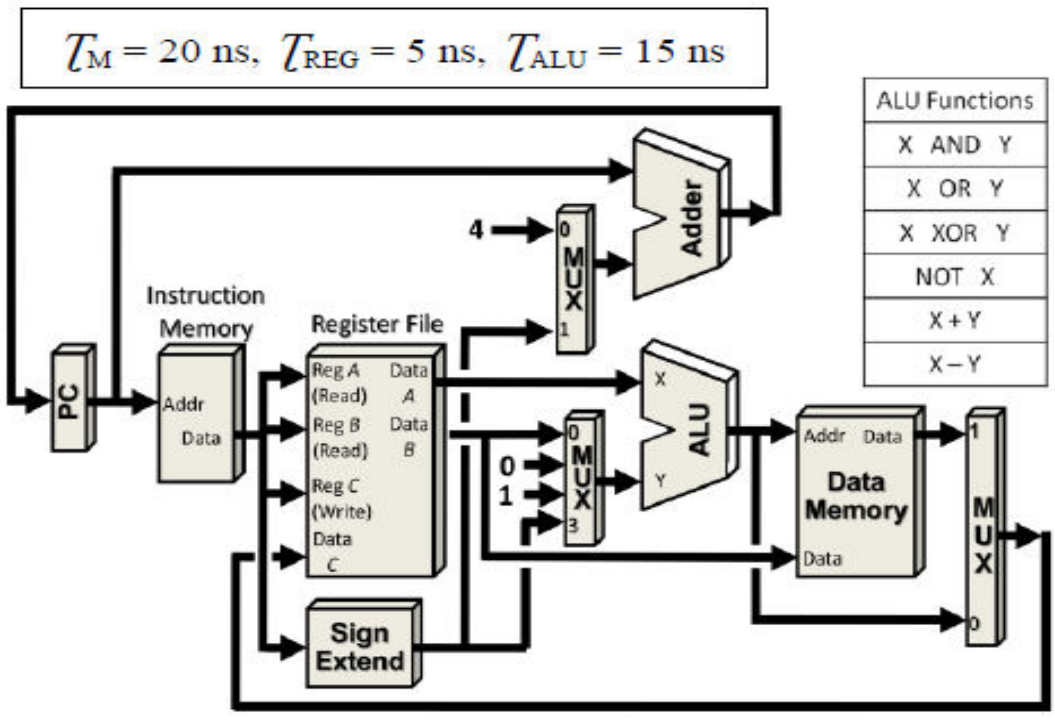
a) Data A ve Data B değerleri nelerdir?

b) Bu saat çevrimi sırasında ALU hangi işlemi yapmaktadır?

c) ALU girişlerinin değerleri nelerdir?

d) Bu komut için minimum saat peryodu (T_{CLK}) nedir?

-



6) Aşağıda özellikleri verilmiş 2 farklı Cache bellek için a-d şıklarını yanıtlayınız?

Cache 1 :

Türü: Direct-mapped cache.

Cache satır uzunluğu (line) 1 byte.

Index = 10-bit; Takı (tag): 6-bit; hit time = 1 cycle

Cache 2:

Türü: 2 way set associative cache.

Cache satır uzunluğu (line) 1 word=4 byte

Index= 7-bit; Takı (tag): 7-bit; hit time = 2 cycle

a) Her 2 bellek için Cache Size (kapasitesini) hesaplayınız?

b) Takıları saklamak için ne kadarlık Cache alanına ihtiyaç bulunmaktadır (her 2 Cache için ayrı ayrı hesaplanacak)?

c) Cache vuru oranı (hit rate) 70 % ve 50% olduğu bilindiği, ancak hangi cache hangi vuru oranına sahip olduğu bilinmediğini kabul ediniz.
Sizce Cache 1 ve Cache 2 için Vuru oranlarını belirlemek mümkünmüdür, nedenini belirterek yazınız?

d) Her 2 cache için ıska zamanı=20 cycle ise ve c) şıkkında belirlenen Vuru Oranları kullanılarak her 2 bellek için Efektif Bellek Varış Zamanını $\mathbf{T_E}$ bulunuz?

7) a) Bir I/O Arabirim Ünitesinin temel görevlerini (fonksiyonlarını) maddeler halinde yazınız?

b) I/O ünitesinin blok diyagramını çiziniz?

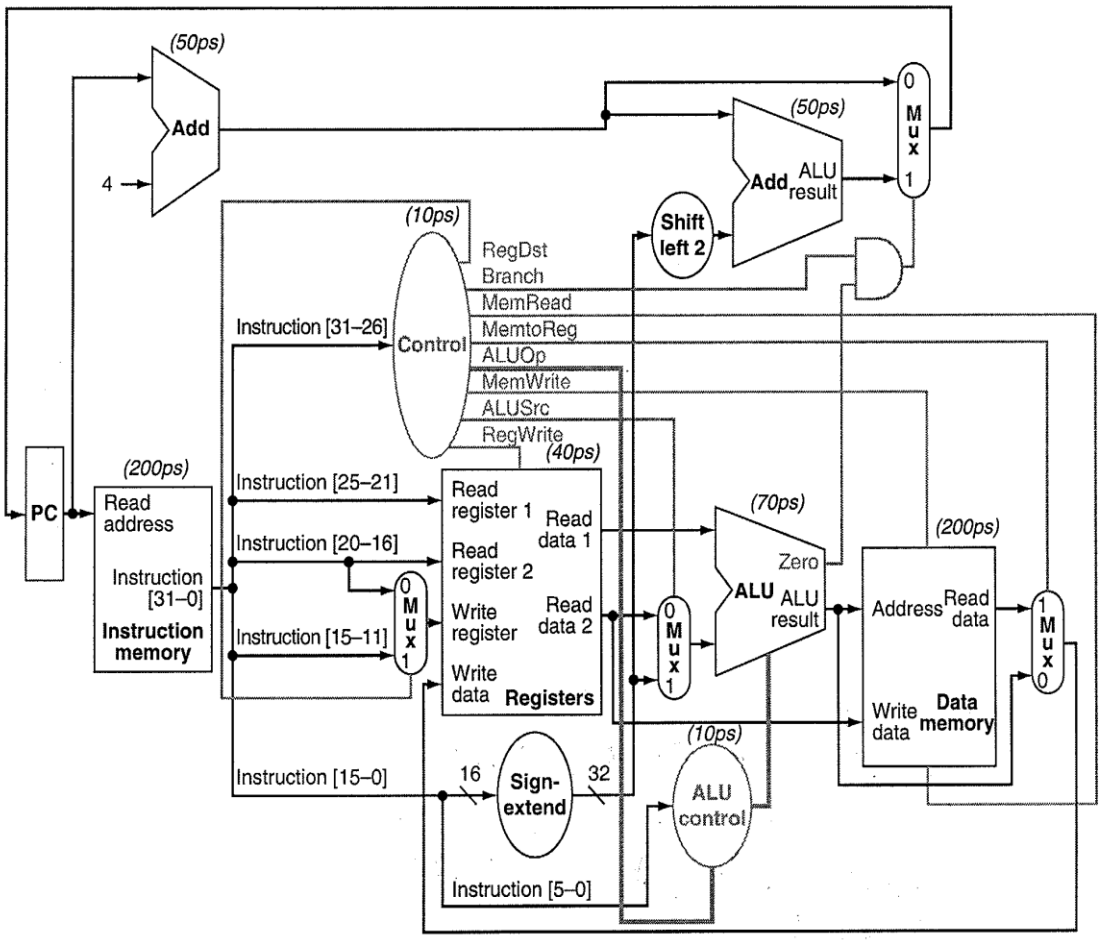
c) Asenkron data transfer yöntemlerini kısaca açıklayınız?

8) Mips komut setine addm komutu ilave edilmek isteniliyor..

addm rd, SBT (rs) ; rd = M[r(s)] + SBT

a) (10p) Bu işlemi yapabilmek için datapath yapısına eklenecek komponentleri ve kontrol çıkışı olarak ekstra ilaveleri de ekleyerek MIPS datapath üzerinde gösteriniz?

b) (10p) Aşağıda verilen Tek Çevrimli MIPS mimarisi için verilen unite gecikmeleri gözönüne alınarak mux ve kontrol işaret gecikmeleri ihmal edilirse normal **add** komutu ve ilave edilen **addm** komut gecikmelerini bulunuz?

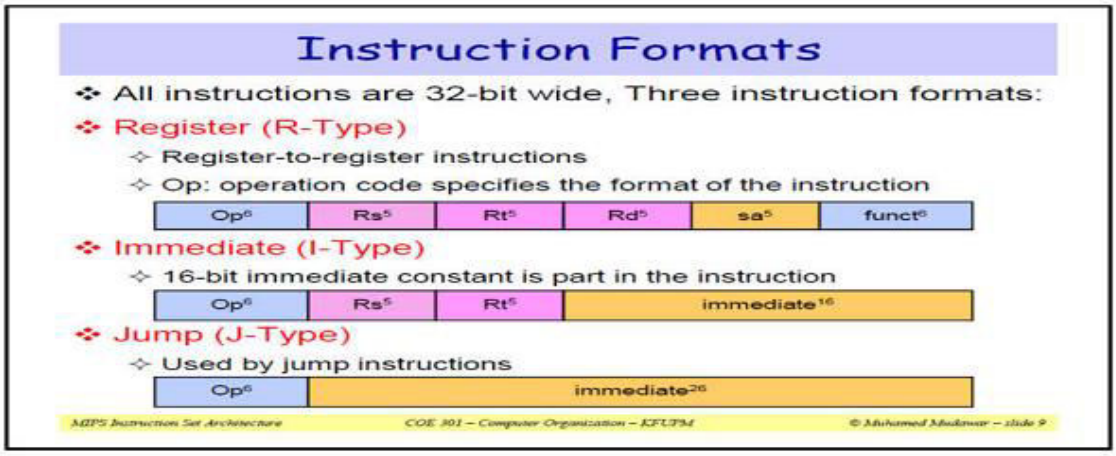


9) a) SRT bölme algoritmasını (radix-2) bölme bitini seçme kuralını tanımlayınız?
b) X=63/256 bölünen sayısını D=9/16 bölen sayısı ile bölüm işlemini yaparak Q=? bölüm bitlerini belirleyiniz?

MIPS Instruction Set Summary (Subset)

Opcodes	Example Assembly	Semantics
add	`add $1, $2, $3`	`$1 = $2 + $3`
sub	`sub $1, $2, $3`	`$1 = $2 - $3`
add immediate	`addi $1, $2, 100`	`$1 = $2 + 100`
add unsigned	`addu $1, $2, $3`	`$1 = $2 + $3`
subtract unsigned	`subu $1, $2, $3`	`$1 = $2 - $3`
add imm. Unsigned	`addiu $1, $2, 100`	`$1 = $2 + 100`
multiply	`mult $2, $3`	`hi, lo = $2 * $3`
multiply unsigned	`multu $2, $3`	`hi, lo = $2 * $3`
divide	`div $2, $3`	`lo = $2/$3, hi = $2 mod $3`
divide unsigned	`divu $2, $3`	`lo = $2/$3, hi = $2 mod $3`
move from hi	`mfhi $1`	`$1 = hi`
move from low	`mflo $1`	`$1 = lo`
and	`and $1, $2, $3`	`$1 = $2 & $3`
or	`or $1, $2, $3`	`$1 = $2 | $3`
and immediate	`andi $1, $2, 100`	`$1 = $2 & 100`
or immediate	`ori $1, $2, 100`	`$1 = $2 | 100`
shift left logical	`sll $1, $2, 10`	`$1 = $2 << 10`
shift right logical	`srl $1, $2, 10`	`$1 = $2 >> 10`
load word	`lw $1, $2(100)`	`$1 = ReadMem32($2 + 100)`
store word	`sw $1, $2(100)`	`WriteMem32($2 + 100, $1)`
load halfword	`lh $1, $2(100)`	`$1 = SignExt(ReadMem16($2 + 100))`
store halfword	`sh $1, $2(100)`	`WriteMem16($2 + 100, $1)`
load byte	`lb $1, $2(100)`	`$1 = SignExt(ReadMem8($2 + 100))`
store byte	`sb $1, $2(100)`	`WriteMem8($2 + 100, $1)`
load upper immediate	`lui $1, 100`	`$1 = 100 << 16`
branch on equal	`beq $1, $2, Label`	`if ($1 == $2) goto Label`
branch on not equal	`bne $1, $2, Label`	`if ($1 != $2) goto Label`
set on less than	`slt $1, $2, $3`	`if ($2 < $3) $1 = 1 else $1 = 0`
set on less than immediate	`slti $1, $2, 100`	`if ($2 < 100) $1 = 1 else $1 = 0`
set on less than unsigned	`sltu $1, $2, $3`	`if ($2 < $3) $1 = 1 else $1 = 0`
set on less than immediate	`sltui $1, $2, 100`	`if ($2 < 100) $1 = 1 else $1 = 0`
jump	`j Label`	`goto Label`
jump register	`jr $31`	`goto $31`
jump and link	`jal Label`	`$31 = PC + 4; goto Label`

EK MIP KOMUT FORMATLARI



Çok Çevrimli MIPS Komut İşleme Aşamaları

Step name	Action for R-type instructions	Action for memory-reference instructions	Action for branches	Action for jumps
Instruction fetch	IR <= Memory[PC] PC <= PC + 4			
Instruction decode/register fetch	A <= Reg [IR[25:21]] B <= Reg [IR[20:16]] ALUOut <= PC + (sign-extend (IR[15:0]) << 2)			
Execution, address computation, branch/jump completion	ALUOut <= A op B	ALUOut <= A + sign-extend (IR[15:0])	if (A == B) PC <= ALUOut	PC <= {PC [31:28], (IR[25:0]),2'b00)}
Memory access or R-type completion	Reg [IR[15:11]] <= ALUOut	Load: MDR <= Memory[ALUOut] or Store: Memory [ALUOut] <= B		
Memory read completion		Load: Reg[IR[20:16]] <= MDR		

0	$zero constant 0 (Hdware)
1	$at reserved for assembler
2	$v0 expression evaluation &
3	$v1 function results
4	$a0 arguments
5	$a1
6	$a2
7	$a3
8	$t0 temporary: caller saves
. . .	(callee can clobber)
15	$t7
16	$s0 callee saves
. . .	(caller can clobber)
23	$s7
24	$t8 temporary (cont'd)
25	$t9
26	$k0 reserved for OS kernel
27	$k1
28	$gp pointer to global area
29	$sp stack pointer
30	$fp frame pointer
31	$ra return address (Hdware)