05.03.2009 tarihli uygulama notları

SORU1:

Bellekte bulunan 8 elemanlı bir dizinin elemanlarını ARTTIR alt programı içinde arttırarak bellekteki gerekli güncellemeleri gerçekleyen bir program RISC1 makinasının dilinde yazılacak ve RISC-1 benzetim programı üzerinde çalıştırılarak benzetim programı anlatılacaktır. Bu sırada program yazımı sırasında derleyicide oluşabilecek hatalar göz önünde bulundurularak gerekli çözümler de üretilecektir.

```
;DIZI nin tan}mlanmas}:

DIZI EQU $500

ORG $500

DCINT 8
DCINT 98
DCINT 45
DCINT 123
DCINT 509
DCINT 345
DCINT 230
DCINT 47


ORG $600

START:
      ADD   R0 , DIZI, R10                        ; R10 <-- #DIZI
      ADD   R0 , 8, R11                     ; R11 <-- 8  (n)
      CALL  ARTTIR (R0), R12                ; R12 <-- PC
                                            ; PC  <-- ARTTIR
                                            ; CWP <-- CWP - 1
;NOT: CALL komutu, komut tablosunda CALL R12, ARTTIR (R0) -eklinde

      NOP


ORG $700

ARTTIR:

      ; DIZI adresi ana programda R10 = R26  (ATTIR da)
      ; n say}s}    ana programda R11 = R27  (ARTTIR da)
      ; donus adresi ana programda R12 = R28 (ARTTIR da)

      XOR R10, R10, R10               ; R10 s}f}rlan}r
                                      ;(dizi elemanlar}na erismede kullan}lacak)
      NOP


TARA:
      LDL (R26)R10, R11               ; R11 e s}radaki eleman al}n}r
      NOP
      ADD R11, 1, R11

      STL (R26)R10, R11
      ADD R10, 4, R10           ; bir sonraki eleman} okumak icin 4 artt}r}ld}
      SUB R27, 1, R27

      ; JMP oncesi 1 adet NOP, dallanma olunca i~ hatt} bo~alt}lacap} igin?
      NOP

      JMP BNE, TARA(R0)               ; n eleman bitmediyse taramaya devam et
      NOP

      RET   (R28)0                    ; PC  <-- (R28) + 0
                                      ; CWP <-- CWP + 1
```

SORU 2:

Berkeley RISC1 mimarisinde I, A, E ile simgelenen 3 segmanlı bir komut işleme hattı vardır.

a) Bu sistemde komut alma ile operand alma/yazma aynı zaman diliminde nasıl gerçekleştirilir?

b) Berkeley RISC1'de yürütülmek için oluşturulmuş aşağıdaki program bölümünde ne tür hatalar vardır?

R14 ve R15 saklayıcılarına önceden gerekli verilerin yüklendiği varsayımı altında bu program bölümü sizce hangi işlevi gerçekleştirmek üzere yazılmıştır?

```
            LDL     (R25)#0,R16
            SUB     R16,R14,R14
            ADD     R25, #1,R25
            LDL     (R25)#0,R17
            SUB     R25, #1,R25
            SUBC    R17,R15,R15
            JMP     MINUS,(R26)#20
            ....
            ....
[(R26)+20]  SLL     R5,#2, R6
```

c) Önceki şıkta önerdiğiniz gerçek işlevi hatasız yerine getirmesi için, bu program bölümünü, sadece satırlarının yerini değiştirerek ve gerekirse araya NOOP komutları yerleştirerek yeniden oluşturunuz. Yaptığınız her değişikliğin nedenini açıklayın ve veri bağımlılığı durumlarını inceleyin.

ÇÖZÜM:

a) Komut ve veri bellekleri ayrı olmalıdır ve bu bağımsız belleklere erişim için ayrı adres/veri yolları bulunmalıdır.

Diğer bir olanak ise komutları önceden çeken bir kuyruk sisteminin oluşturulmasıdır.

b) Bu programda 64 bitlik iki sayı arasında çıkarma yapılıp (önce düşük anlamlı 32 bir daha sonra yüksek anlamlı 32 bit eldeli/borçlu olarak çıkartılıyor) eğer sonuç negatifse bir adrese dallanılmaya çalışılıyor.

Programda mantıksal hatalar ve iş hattı yapısından kaynaklanan hatalar bulunmaktadır.

```
            LDL     (R25)#0,R16
            SUB     R16,R14,R14     veri bağımlılığı
            ADD     R25, #1,R25     +4
            LDL     (R25)#0,R17
            SUB     R25, #1,R25     elde (borç) kayboluyor
            SUBC    R17,R15,R15
            JMP     MINUS,(R26)#20  Dallanma gecikmesi
            ....
            ....
[(R26)+20]  SLL     R5,#2, R6
```

c) Programın düzeltilmiş hali aşağıda verilmiştir. Satırların yer değiştirmesi problemleri çözmektedir. NOOP eklemeye gerek yoktur.

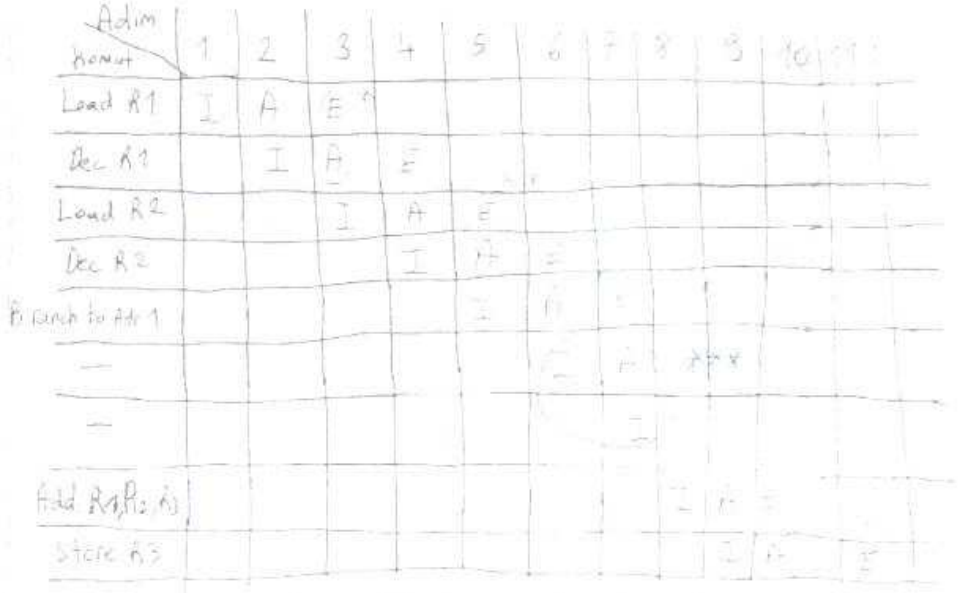
```
            LDL     (R25)#0,R16
```

```
                ADD         R25, #4,R25
                LDL         (R25)#0,R17
                SUB         R16,R14,R14
                SUBC        R17,R15,R15
                JMP         EQUAL,(R26)#20
                SUB         R25, #4,R25
                ...
[(R26)+20]      SLL         R5,#2, R6
```

SORU3:

Derste incelenen Berkeley RISC I makinasının komut işleme (pipeline) I, A, ve E segmanlarından oluştuğu hatırlatılır. Aşağıdaki komut dizisi bu makine tarafından yürütüldüğünde ortaya çıkabilecek sorunlar nelerdir? Bu sorunlar en iyi şekilde derleyici tarafından nasıl ortadan kaldırılırlar?

```
                LOAD R1
                DEC R1
                LOAD R2
                DEC R2
                BRANCH TO ADR1
                .
                .
                .
ADR1    ADD R1,R2,R3
        STORE R3
```



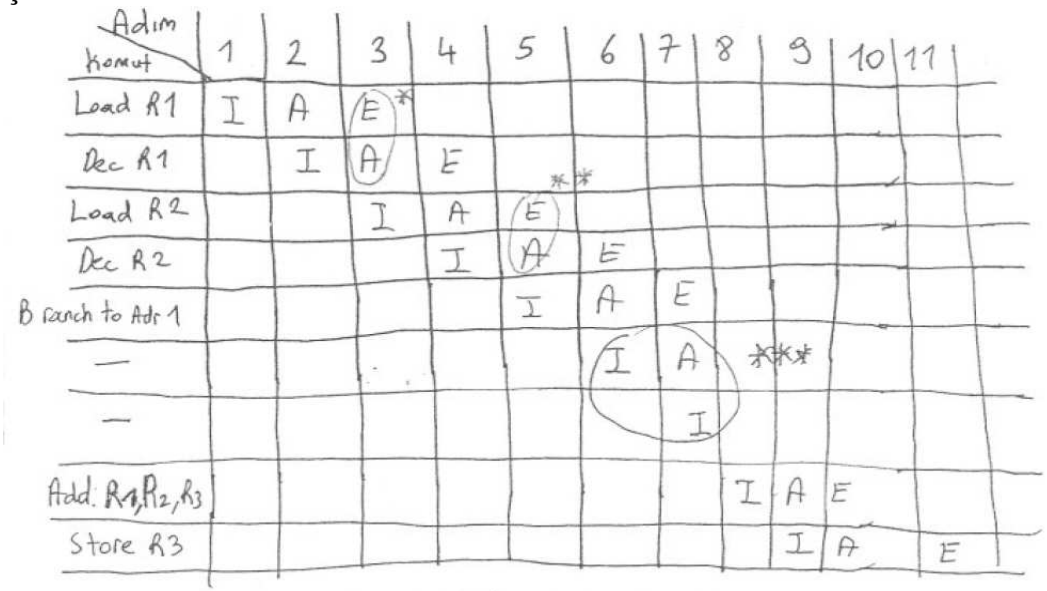
Veriye bağımlılık (Data dependency) sorunu yaşanır.

SORU. 3

Derste incelenen Berkeley RISC 1 makinasının komut iş hattının (pipeline) I, A, ve E segmanlarından oluştuğu hatırlatılır. Aşağıdaki komut dizisi bu makine tarafından yürütüldüğünde ortaya çıkabilecek sorunlar nelerdir? Bu sorunlar en iyi şekilde derleyici tarafından nasıl ortadan kaldırılırlar?

```
            LOAD R1
            DEC R1
            LOAD R2
            DEC R2
            BRANCH TO ADR1
                    -
                    -
                    -
ADR1        ADD R1,R2,R3
            STORE R3
```

Çözüm:



1) *, ** Veriye bağımlılık (Data dependency) sorunu yaşanır.

2) *** Dallanma komutundan sonraki 2 komut iş hattına boş yere alınmış olur.

Programın derleyici tarafından en iyi şekilde düzeltilmiş hal:

```
            Load R1
            Load R2
            Branch to adr1
            Dec R1
            Dec R2
            =
ADR1        Add R1,R2,R3
            STORE R3
```

SORU 4:

$X_i = A_i \cdot X_{i-k} + B_i$ rekürans bağıntısını hesaplayan bir "pipeline" yapı tasarlanacaktır. k "pipe" segman sayısıdır. A_i ve B_i bellekten aynı zaman diliminde okunabilmektedir. Sonuç bir taraftan ayrı bir belleğe kaydedilirken diğer yandan rekürans hesabı için kullanılmaktadır. İşlem başladığında yapıdan sonuç elde edene kadar X girişinin "0" değer aldığı varsayılacaktır. Yapıyı tasarlamak için gerekli sayıda çarpma, toplama elemanı ile saklayıcı kullanılabilir.

Bellek erişim süresi : 45 ns

Toplayıcı işlem süresi: 40 ns.

Çarpıcı işlem süresi: 55 ns.

Saklayıcı gecikmesi: 5 ns.

a) Yapıyı tasarlayıp çiziniz ve ilk 6 adım için segmanlardaki veri akışını bir tablo halinde gösteriniz.

b) Yapının saat periyodunu belirledikten sonra "pipeline" kullanılmasının getirdiği hızlanmayı hesaplayınız.

Çözüm:

1.katman bellekten A_i ve B_i'lerin okunması, X_i değerinin yüklenmesi

2. katman $A_i * X_{i-k}$ işlemi

3. katman $A_i * X_{i-k} + B_i$ işlemi

Gecikmeler:

1. 45 + 5 =50 ns
2. 55 + 5 = 60 ns
3. 40 + 5 = 45 ns

İşhattı yapısı kullanılmazsa bir adımdaki gecikme: Ts = 50 + 60 + 45 = 155ns

İş hattı yapısı kullanılırsa bir adımdaki ortalama gecikme Tp = 60 ns

Bu durumda Hızlanma:

S = Ts/Tp = 155/60 = 2.58