MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ
2019-2020 Öğretim Yılı Bahar Yarıyılı, Bilgisayar Organizasyonu Dersi
Final Sınav Ödevi

Öğretim Elemanı Unvanı / Adı Soyadı: Prof.Dr. İsmail KADAYIF Veriliş Tarihi: 01/06/2020

Öğrenci Adı Soyadı: Öğrenci No: Teslim Tarihi: 08/06/2020

Not: Soruların cevapları ilgili boşluklara yazılacaktır (type edilecek, elle yazılmayacak). Sonra dosyanız pdf dosyasına dönüştürerek UBYS sistemine yüklenmelidir.

1) Aşağıdaki soruları cevaplayınız.

a) 32 bitlik ikinin tümleyeni yönteminde ifade edilebilecek en büyük tam sayıyı hexadecimal (on altılık) yöntemde yazınız. **(5 puan)**

0111 1111 1111 1111 1111 1111 1111 1111

= 0x7FFFFFFF

b) 32 bitlik ikinin tümleyeni yönteminde ifade edilebilecek en küçük tam sayıyı hexadecimal (on altılık) yöntemde yazınız. **(5 puan)**

1000 0000 0000 0000 0000 0000 000 0000

= 0x8000000

c) -250.125 kayan noktalı sayıyı tek duyarlı (single precision) yöntemde yazınız. **(10 puan)**

Single precision dedigi icin 32 bit uzerinde calisacagiz. Sayi negatif dolayisiyla sign bit = 1. s=1

$(250)_{10}= (11111010)_2$. $(0.125)_{10}= (.001)_2$

Sayiyi bilimsel binary notasyonda gosterirsek: 11111010.001 oluyor. En bastaki 1'i gizli bir yapiyorduk, dolayisiyla noktayi 7 birim sola aliyoruz: 1.1111010001×2^7 oluyor sayimiz. Dolayisiyla e=7.

7+127=134'un binary hali, 10000110 exponent kismimiz olcak.

111 1010 0010 0000 0000 0000 da fraction kismi oluyor.

1 1000 0110 (1)111 1010 0010 0000 0000 0000 = 0xc37a2000 (sol bastan saga dogru 4erli gruplari hex'e cevirerek yazdik.

2)

Yukarıda tanımlamanın 15x10 boyutlarında iki boyutlu bir tam sayı dizisine ait olduğunu düşünelim. Ayrıca bu dizinin belleğe row-major yöntemine göre yerleştirildiği ve dizi[5][3] elemanının adresinin 0x10010A80 olduğu bilinmektedir. Buna göre;

a) dizi[7][8] elemanının adresini bulunuz. **(5 puan)**

En son hucre dizi[14][9]. Row majorda satirlar sira sira. Eger adres 0'dan baslamis olsaydi dizi[5][3]: 4x(5x10+3) = 212 byte => 0x000000D4 olurdu. Dolayisiyla dizi[5][3]un gercek adresinden 0x000000D4'u cikartarak dizi[0][0] i buluruz.

0x10010A80 – 0x000000D4 = 0x100109AC = dizi[0][0]

dizi[7][8] de 312 byte ileride oldugundan(0x00000138): 0x100109AC + 0x00000138 = 0x10010AE4 = dizi[7][8]

b) Bu dizinin başlangıç adresini (dizi[0][0] ın adresi) bulunuz. **(5 puan)**
a) şıkkında bulmuştuk. 0x100109AC = dizi[0][0]

c) Bu dizinin en son elemanının adresini bulunuz. **(5 puan)**
En son hucre dizi[14][9]. Son elemanin byte'i: 4x(14x10+9)= 596
0x00000254 ilerde => 0x100109AC + 0x00000254 = 0x10010C00 = dizi[14][9]

3) Aşağıdaki soruları cevaplayınız?

a) jal (jump and link) ve jr (jump register) komutlarının temel farkını hangi durumlarda kullanıldıklarını dikkate alarak kısaca açıklayınız. **(5 puan)**

jal (jump and link) sartsiz bir adrese ziplamadir, verilen adresteki ziplamadan hemen sonraki komutun adresini $ra($31) [return address]'e kaydeder. Bu sayede alt program ile cagrinin kaynagi birbirine baglanmis olur.

jr (jump register) $ra (return address)'nın içerdiği adresi PC (Program Counter)' ye yazar yani $ra nın işaret ettiği adrese sıçranır. Ozetle verilen adrese sicrar.

b) add $4,$5,$6 komutunun makine dilinde kodlanmasını hexadecimal düzende gösteriniz. **(5 puan)**

Once $5 sonra $6, sonra da $4 yazilir =>

add D, S ,T = 0000 00 ss ssst tttt dddd d000 0010 0000

add $4, $5, $6 => 4 = 00100, 5 = 00101, 6 = 00110

0000 0000 1010 0110 0010 0000 0010 0000 = 0x00A62020

c) addi $8,$8,77 komutunun makine dilinde kodlanmasını hexadecimal düzende gösteriniz. **(5 puan)**

Once sagdaki $8 sonra soldaki, en son da 77 yazilir =>

addi T,S,I = 0010 00ss ssst tttt iiii iiii iiii iiii

addi $8, $8, 77 => $8 = 01000, 77 = 0000 0000 0100 1101

0010 0001 0000 1000 0000 0000 0100 1101 = 0x2108004D

d) Ekrana veri yazma, konsoldan veri okuma gibi çevre birimlerine erişmek istenildiğinde bu eylemle ilişkili işletim sistemi yordamı çağrılmaktadır. Bu tip işlemlerin işletim sistemi yordamına yaptırılmasının önemli sebeplerinden iki tanesini yazınız. **(10 puan)**

Bu tip islemleri isletim sistemin kendi tanimlanmis print ve input komutlarina yaptirmak sart yoksa her programlama dili icin sistemde farkli bir print, farkli bir input komutunun tanimlanmis olmasi gerekirdi. Assembly bunlari tek bir komuta indirgiyor CPUnun isleyebilmesi icin.

Öğretim Elemanı Unvanı / Adı Soyadı: Prof.Dr. İsmail KADAYIF Veriliş Tarihi: 01/06/2020

Öğrenci Adı Soyadı: Öğrenci No: Teslim Tarihi: 08/06/2020

4)

```
a:      .word
c:      .word
d:      .word
e:      .word
sonuc:  .word
```

Yukarıdaki tanımlama verilmektedir. **sonuç = a + c – d x e** işlemi yapılmak istenmektedir. Sistemde sadece $s0 ve $s1 olmak üzere iki tane genel amaçlı yazmaç ve $sp, HI ve LO olmak üzere üç tane de özel amaçlı yazmaç olduğu bilinmektedir. $sp yazmacının stack göstergesi olduğu ve sadece verileri stacke itmek için kullanılabileceğini ve genel aritmetik işlemlerde kullanılamayacağını, HI ve LO yazmaçlarının ise sadece çarpma işleminin sonucunu saklayan yazmaçlar olduğunu ve bunlara kullanıcı tarafından herhangi bir değer yüklenemeyeceğini göz önünde bulundurarak yukarıdaki işleme ait kodu QtSpim simülatöründe çalışacak şekilde yazınız. **Not**: Basitlik olması açısından **d x e** işleminin sonucunun 32 bite sığdığını kabul ediniz. **(20 puan)**

```
.data
a: .word 3
c: .word 5
d: .word 7
e: .word 10
sonuc: .word 0          # qtspimde calistirabilmek icin
.text
.globl main
main:
        lw $s0,d            # d tanimlamasi
        lw $s1,e            # e tanimlamasi
        mult $s0, $s1       # d * e
        mflo $s0            # move from Lo
        addi $sp, $sp, -4   # stack pointer 1 birim azaltilir
        sw $s0, ($sp)
        lw $s0, a           # a tanimlamasi
        lw $s1, c           # c tanimlamasi
        add $s0, $s1, $s0   # a + c
```

```
addi $sp, $sp, -4
sw $s0, ($sp)
lw $s0, ($sp)          # stack pointerdaki degeri (a+c) s0 yazmacina yazdik
addi $sp, $sp, 4
lw $s1, ($sp)          # stack pointerdaki degeri (d*e) s1 yazmacina yazdik
addi $sp, $sp, 4
sub $s0, $s0, $s1      # (a+c) - (d*e)
lw $s1, sonuc          # sonucu $s1 yazmacina yazdik.
add $s1, $s1, $s0      # s1 yazmacina stack pointer degerini ekledik
li $v0, 10             # programi hatasiz bitirmek icin komut + syscall
syscall
```

Öğretim Elemanı Unvanı / Adı Soyadı: Prof.Dr. İsmail KADAYIF Veriliş Tarihi: 01/06/2020

Öğrenci Adı Soyadı: Öğrenci No: Teslim Tarihi: 08/06/2020

5) C dilinde yazılı aşağıdaki kod parçacığı verilmektedir. Amaç, bu kod parçacığına karşı gelen kodu QtSpim simülatöründe çalıştırılmaktır.

```
int x;
int *ptr;
  dizi:    .space 600
x = 22;
ptr = &x;
*ptr = 100;
```

a) Yukarıdaki veri tanımlamalarına karşı gelen veri tanımlamalarını QtSpim için yazınız. **(5 puan)**

```
.data

x :             .word 0

ptr :           .word 0
```

b) Yukarıdaki C kouna karşı gelen kodu QtSpim için yazınız. **(15 puan)**

```
.data

x :             .word 0

ptr :           .word 0

        .text
        .globl main
        main:
        la $s0, x               # xi s0 yazmacina yerlestirdik
        la $s1, ptr             # ptri s1 yazmacina yerlestirdik
        li $t0, 22              # 22yi t0 yazmacina yerlestirdik
        sw $t0, ($s0)           # t0 yazmacindaki 22yi s0 yazmacindaki xin icine yazdik
        li $t0, 100             # t0 yazmacina 100 yazdik
        sw $s0, ($s1)           # ptr = &x
        sw $t0, ($s0)           # *ptr = 100
        li $v0, 10              # programi bitirmek icin komut ve syscall
        syscall
```