1986

### DOĞU AKDENİZ ÜNİVERSİTESİ

**BİLGİSAYA MÜHENDİSLİĞİ ANABİLİMDALI**

**İlkbahar 2019-20**

# CMPE 325 - Bilgisayar Mimarisi II

**DENEY 2:**

**Dizilere Erişimde Kullanılan Bellek Referanslı MIPS Talimatları**

**Amaç:**

Fonksiyonel tek modül birleştirici programlarında Veri Segmenti ve Dizilerin kullanımına giriş. Değişken sembollerle anlaşılabilir montaj programlarının geliştirilmesi.

# Giriş:

Üst düzey dillerdeki programların aksine, MIPS düzeneğindeki aritmetik komut dizileri değişken olamaz. Sınırlı sayıda genel amaçlı kayıttan olmalıdırlar. MIPS mimarisinde 32 g.p. kayıtçı var (0 $, 1 $, 2 $,…., 31 $). Kayıtların bir programlama dilinin değişkenlerinden ana farklarından biri sınırlı sayılarıdır. Kısıtlı kayıt sayısı, içeriğini okumak ve yazmakta daha yüksek hız sağlar. Kayıtların sayısı arttıkça, adres kod çözme işleminde daha fazla geçit gerekir, bu da daha yüksek gecikmelere neden olur ve bu nedenle adres kod çözme kapılarındaki gecikmeler nedeniyle saat döngü süresini yükseltir.

Çoğu programlama dilinde, tek veri elemanları içeren basit veri tipleri ve ayrıca diziler gibi daha karmaşık veri yapıları vardır. Bu karmaşık veri yapıları, işlemcinin mevcut kayıt defterlerinden çok daha fazla veri öğesi içerebilir.

Bir işlemcinin veri depolamak için yalnızca sınırlı sayıda kaydı vardır. O zaman bir işlemci bu kadar büyük yapılara nasıl erişebilir ve işleyebilir?

Cevap, bir işlemci milyonlarca veri öğesi içeren bir hafıza sistemine de erişebiliyor. Dolayısıyla, diziler gibi büyük veri yapıları bellekte tutulur. Kayıtlara yalnızca işlenmesi gereken verilerin bir kısmı yüklenir.

MIPS mimarisi, bir dizi bellek aktarma talimatı sağlar. Bellek referans komutu yükleme sözcüğü (lw) kullanılarak bellekten kayıt cihazına veri aktarımı mümkündür, **load word** (**lw**),

### lw $destination, offset($base)

Bu **$destination** hedefini, bellek adres ofsetinden başlayarak veri kelimesiyle yükler. **offset+$base**.

Örneğin, **lw $20, 10000 ($0)** adresindeki bellek konumundaki sabit **10000+$0** kayıt defterinde saklandığı anlamına gelir. **$20**.

Bir sicilden bir belleğe veri aktarımı,**store word** (**sw**) komutu kullanılarak elde edilir,

### sw $source, offset($base)

Hangi **$source** başlayan adrese belleğe kaydeder. **offset+$base**.

Örneğin, **sw $10, 20000($2)** kayıt içeriğini saklar. **$10**

adresindeki hafıza konumuna girin. **20000+$2**.

Hem **lw** hem de **sw** komutlarında, **offset is a 16-bit signed integer**.

Ancak, sözde talimatlara izin verilirse, 32 bit adresleri kabul ederler ve yalancı komutu kullanarak **lui**, **add**, ve **sw** veya **lw** komutlarını kullanarak 16-bit-offset kullanılır.

# Deneysel Çalışma:

## Bölüm 1 – İzleme Egzersizi

Bellek referans talimatlarının (lw ve sw) çalışmasını anlamak için, aşağıdaki "**exp2a.s**". dosya adını veren aşağıdaki MIPS montaj programını yazıp uygulayın. Program **la $dest,<label>**

gibi sahte talimatlar içeriyor.

ve lw ve sw komutlarında 32 bit ofset var. Bu nedenle, SPIM’in sözde talimatı seçeneğini, ayarlar iletişim kutusundan **alt-s,s**). açmanız gerekir. Veri setinize asistanınızdan bilgi verin ve programınızı bu veri setiyle izledikten sonra raporun ilgili bölümünü doldurun. Yürütmeyi bu adresten başlatmak için PC'deki değeri 0x00400000 olarak ayarlamayı unutmayın.

**.data 0x10000000**

**# A[] is array with a trailing zero. A: .word 128,100,42,16,5,2,0,0**

**Count:**

**.word 0 Sum:**

**.word 0 Result:**

**.word 0 Remainder:**

**.word 0 EndofData:**

**.word -1**

**# End of data segment, code starts here from address 0x00400000 # Turn-off bare-machine, and turn-on pseudo-instruction options.**

**# for the sake of SPIM add below one empty line**

**# move from LO to $destination # move from HI to $destination # mean**

**# and this is the reminder of division**

**syscall**

**# quotient is in LO, reminder is in HI**

**div $10,$8 mflo $11 mfhi $10**

**sw $11,Result($0)**

**sw $10,Remainder($0)**

**# save the count**

**sw $8,Count($0) sw $10,Sum($0)**

**# divide $10 by $8, use count of repeating subtractions.**

**# A[0]+...+A[i] -> $2**

**# ++count**

**# address of(A[i+1]) -> $2**

**# loop until getting the zero**

**slp**

**j slx:**

**addi $8,$8,1 addi $2,$2,4**

**# A[i] -> $11**

**$11,0($2)**

**$11,$0,slx**

**$10,$10,$11**

**# pseudo-instruction load-address A[.] # 0 -> $8 , count**

**# 0 -> $10, sum**

**$2,A**

**$8,$0,$0**

**$10,$0,0**

**la or or**

**slp:**

**lw beq add**

**.text**

**.globl main main:**

**# finding average of A[]**

## Bölüm 2 – Proglama Alıştırması

1. [A] 'nın dört kelimenin (20,40,60,80) toplam sözcüklerinin toplam değerine oranını hesaplamak için bir program yazın ve ardından dört oran sonucunu [res] hafıza konumuna kaydedin. Programınızda hata ayıkladıktan sonra, programınızı verilen belirli verilerle takip edin ve veri segmentini rapor sayfanıza doldurun.

res[0]180/20 res[1]180/40 res[2]180/60 res[3]180/80

.data 0x10000000 A:

.word 20,40,60,80

res:

.word 0,0,0,0

.text

.globl main main:

.

.

COMPLETE THE PROGRAM

.

.

.

syscall # for the sake of SPIM add below one empty line

2- Programınızı asistanınızın yönüne göre değiştirin ve veri bölümünü rapor sayfanıza doldurun

İsim: Öğrenci Numarası:

Gönderen (Yrd.): Tarih:dd/mm/yy / /

1986

### DOĞU AKDENİZ ÜNİVERSİTESİ

**BİLGİSAYA MÜHENDİSLİĞİ ANABİLİMDALI**

**İlkbahar 2019-20**

# CMPE 325 - Bilgisayar Mimarisi II

### DENEY 2 - Rapor Formu

**Bölüm1:** Veri Segmenti Satırlarınız

**A: .word , , , , , , ,**

**:**

**.word 0**

**:**

**.word 0**

**:**

**.word 0**

**:**

**.word 0 EndofData:**

**.word -1**

İdamdan sonra Onaltılık veri segmenti okumaları

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **0x10000000** |  |  |  |  |  |
| **0x10000010** |  |  |  |  |  |
| **0x10000020** |  |  |  |  |  |
| **0x10000030** |  |  |  |  |  |
| **0x10000040** |  |  |  |  |  |
| **0x10000050** |  |  |  |  |  |
| **....** |  |  |  |  |  |

**Bölüm2:** Programlama Alıştırması

1. Veri kümeniz:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **0x10000000** |  |  |  |  |  |
| **0x10000010** |  |  |  |  |  |
|  |  |  |  |  |  |

1. Değişiklikten sonra:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **0x10000000** |  |  |  |  |  |
| **0x10000010** |  |  |  |  |  |
| **0x10000020** |  |  |  |  |  |
| **0x10000030** |  |  |  |  |  |
| **0x10000040** |  |  |  |  |  |
| **0x10000050** |  |  |  |  |  |
| **....** |  |  |  |  |  |

Derecelendirme:

Sınav Performansı:

LabaratuvarPerformansı:

Yrd. Gözlemler: