DOĞU AKDENİZ ÜNİVERSİTESİ BAHAR 2012-2013

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

BLGM-324 BİLGİSAYAR MİMARİSİ

DENEY #3

MIPS Modülar Programlamada Jump-and-Link (jal) ve Jump-Return (jr) Komutlarının Kullanımı

1.Giriş

Programcılar bir programı birimselleştirmek için altyordam kullanır. Böylece program anlaşılması kolaylaşır ve kod parçaları tekrar kullanılabilir. Bir komut takımı program birimleri kurmak için *yordam çağır* ve *yordamdan dön* gibi destekleyici komutlar sağlamalıdır. Ayrıca yordama değiştirgeleri nasıl aktaracağımız ve yordam çağrılarını nasıl iç içe koyacağımız gibi konularda kurallar gerekir. Modülar programlama için MIPS de belirlenmiş iki komut bulunur. Bu komutlar: ***Jump-and-link*** (**jal**) and ***Jump-register*** (**jr**).

```
      ...
      jal   ProcedureAddress
      add   $2,$5,$9 # next-instruction after the call
      ...
ProcedureAddress:
      ...
      jr    $31
AnotherProcedure:
      ...
```

jal YordamAdresi:

jal önce birsonraki yeri göstermek üzere program sayacını arttırır (*PC*←*PC*+4), ardından bu değeri **r31** -e saklar. MIPS -in yordamdan dönüş için dönüş-adresine-atla (*jump-returnaddress* yada *jump to register*) komutu bulunur. Bu komutun anımsatıcısı **jr** -dir, ve dönüş adresini içeren yazmacı belirten bir tek işleneni vardır. **jal** dönüş adresini **r31**-e sakladığından, **jr** genellikle **$31** ile kullanılır:

jr $31

jr $31, yürütürken **r31** -in içeriğini (dönüş adresi) *PC* -ye kopyalar, veböylece **r31** -te saklanan adrese atlamış olur.

Eğer bir prosedür başka prosedürü çağırırsa $31 yazmaçındaki eski değer LIFO(son-gelen-ilk-gider) yığınında kaydedilmelidir.

MIPS komutları parametrelerin nasıl çağrıldığını ve içiçe yordam çağrılarını açıklar. Yazmaçlar özel amaçlar için gruplandırılımıştır.

1. **$0:** 0 içerir, **$1:** çevirici pseudocode için yer ayırır.
2. **$2 ve $3** çağırılandan(callee) çağırana(caller) döndürülen değer yada döndürülen değerin göstergesi için kullanılır.
3. **$4, $5, $6** ve **$7** çağırandan (caller) çağırılana (callee) argümanların aktarılması için kullanılır.
4. **$8…$15** çağrılan (callee) kayıt yazmaçlarıdır.
5. **$16…$23** çağıran (caller) kayıt yazmaçlarıdır.
6. **$28** global very göstergesidir ve static very segmentini gösterir.
7. **$29** yığın-göstergesi ve yığının en üstteki adresini gösterir.

Bu deneyde sadece çağırılan-kayıt ve çevirici pseudo-kodu kullanacağız.

2.Deneysel Çalışma

1. İçiçe yordam çağrıları:

Aşağıdaki C programı MIPS ile yazılmış ve jal ve jr komutlarının nasıl çalıştığını görebilirsiniz. Bu program belirli bir başlangıç indeksinden bitiş indeksine kadar tekrarlanan çağrılarda dizinin toplamını bulur.

```
// Function computes sum of the array elements which starts from first
element (0) and ends when reaches to (size-1)th index

int sum (int arr[ ], int size)
{
        If (size==0)
                return 0;
        else
                return sum(arr, size-1) + arr[size-1];
}
```

Aşağıdaki kod MIPS assembly kodu ile ilgilidir.

```
.data 0x10000000
A: .word 3,5,6,2,0,4
.text 0x00400000
.globl main
main:
la $a0, A
li $a1, 6
jal fun
move $s0, $v0
syscall
fun: addi $sp, $sp, -8                  # Adjust sp
addi $s0, $a1, -1                       # Compute size - 1
sw $s0, 0($sp)                          # Save size - 1 to stack
sw $ra, 4($sp)                          # Save return address
bne $a1, $zero, L1                      # branch ! ( size == 0 )
li $v0, 0                               # Set return value to 0
addi $sp, $sp, 8                        # Adjust sp
jr $ra                                  # Return
L1: move $a1, $s0                       # update second arg
jal fun
lw $s0, 0($sp)                          # Restore size - 1 from stack
li $t7, 4                                # t7 = 4
mult $s0, $t7                           # Multiple size - 1 by 4
mflo $t1                                 # Put result in t1
add $t1, $t1, $a0                       # Compute & arr[ size - 1 ]
lw $t2, 0($t1)                          # t2 = arr[ size - 1 ]
add $v0, $v0, $t2                       # retval = $v0 + arr[size - 1]
lw $ra, 4($sp)                    # restore return address from stack
addi $sp, $sp, 8                         # Adjust sp
jr $ra                                  # Return
```

İlk olarak başlangıç PC adresi **ØxØØ4ØØØØØ** olmalıdır. Programı yükleyin ve programı f10 ile çalıştırın. Yığın-listesinin içeriğini rapora yazınız, ve **jr $31** yazmaçını gördüğünüzde değerini yazınız.

Bölüm-2 Programlama örneği:

Bu deneyde size verilen C kodunu MIPS koduna çeviriniz.

İsim:_____ Öğrenci No:_____

DOĞU AKDENİZ ÜNİVERSİTESİ BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ BLGM-324 BİLGİSAYAR MİMARİSİ

DENEY #3

Bölüm-1: jr $31 komutunu ilk çalıştırdığımızda yığının içeriğini yazınız.

Address		word0	word1	word2	word3
addr.					
register					
addr.					
register					
addr.					
register					
addr.					
register					
addr.					
register					

İçiçe yordam çağrılarının derinliği nedir?

Yığında kaç tane sözcük tutulur?

Bölüm-2: Programlama Ödevi

1. C kodunu MIPS koduna çevirin.
2. Hesaplanan dizi elemanlarının içeriğini aşağıya yazınız (onaltılık tabanda).

..... , ... , ... , ...
..... , ... , ... , ...
..... , ... , ... , ...
..... , ... , ... , ...
..... , ... , ... , ...

Deney Notu: