

Öğrenci Adı – Soyadı: \_\_\_\_\_  
Öğrenci Numarası: \_\_\_\_\_

<i>S1</i>	<i>S2</i>	<i>S3</i>	<i>S4</i>	<i>S5</i>	<i>S6</i>	<i>S7</i>	<i>Toplam</i>



Hacettepe Üniversitesi  
Bilgisayar Mühendisliği Bölümü

**BİL 220**  
Sistem  
Programlamaya Giriş  
Ara Sınav 1

**Tarih:** 29 Mart 2012

**Süre:** 140 dak.

**Sınava başlamadan önce aşağıda yazılanları mutlaka okuyunuz!**

- Bu sınav **kapalı kaynak** bir sınavdır. Yani sınav süresince ilgili ders kitapları veya ders notlarınızdan faydalanmanız yasaktır.
- Size yardımcı olması açısından sonraki 2 sayfada bazı Intel IA32/x86-64 Assembly komutlarının söz dizimleri ve diğer bazı ilgili tanımlar verilmiştir.
- **Sınavda kopya çekmek yasaktır.** Kopya çekmeye teşebbüs edenler hakkında ilgili idare işlemler **kesinlikle** başlatılacaktır.
- Her bir sorunun toplam ağırlığı soru numarasının ardında parantez içinde belirtilmiştir.
- Sınav toplam 125 puan üzerinden değerlendirilecektir.

*Sınav bu kapak sayfası dahil toplam 11 sayfadan oluşmaktadır. Lütfen kontrol ediniz!*

**BAŞARILAR!**

### Sıçrama İşlemleri

Sıçrama	Koşul
jmp	1
je	ZF
jne	~ZF
js	SF
jns	~SF
jg	~(SF^OF) & ~ZF
jge	~(SF^OF)
jl	(SF^OF)
jle	(SF^OF)   ZF
ja	~CF & ~ZF
jb	CF

### Aritmetik İşlemler

Format	İşlem
addl <i>Src, Dest</i>	Dest = Dest + Src
subl <i>Src, Dest</i>	Dest = Dest - Src
imull <i>Src, Dest</i>	Dest = Dest * Src
sall <i>Src, Dest</i>	Dest = Dest << Src
sarl <i>Src, Dest</i>	Dest = Dest >> Src
shrl <i>Src, Dest</i>	Dest = Dest >> Src
xorl <i>Src, Dest</i>	Dest = Dest ^ Src
andl <i>Src, Dest</i>	Dest = Dest & Src
orl <i>Src, Dest</i>	Dest = Dest   Src
incl <i>Src</i>	Dest = Dest + 1
decl <i>Src</i>	Dest = Dest - 1
negl <i>Src</i>	Dest = - Dest
notl <i>Src</i>	Dest = ~ Dest

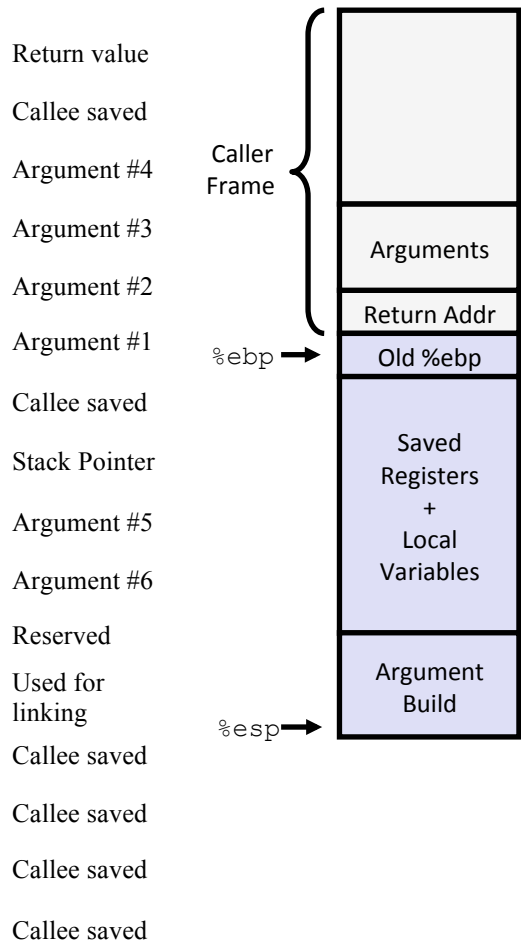
### Bellek İşlemleri

Format	İşlem
(Rb, Ri)	Mem[Reg[Rb]+Reg[Ri]]
D(Rb, Ri)	Mem[Reg[Rb]+Reg[Ri]+D]
(Rb, Ri, S)	Mem[Reg[Rb]+S*Reg[Ri]]

### Yazmaçlar (Registers)

63	31	15	87	0
%rax	%eax %ax	%ah	%al	
%rbx	%ebx %bx	%bh	%bl	
%rcx	%ecx %cx	%ch	%cl	
%rdx	%edx %dx	%dh	%dl	
%rsi	%esi %si		%sil	
%rdi	%edi %di		%dil	
%rbp	%ebp %bp		%bpl	
%rsp	%esp %sp		%spl	
%r8	%r8d %r8w		%r8b	
%r9	%r9d %r9w		%r9b	
%r10	%r10d %r10w		%r10b	
%r11	%r11d %r11w		%r11b	
%r12	%r12d %r12w		%r12b	
%r13	%r13d %r13w		%r13b	
%r14	%r14d %r14w		%r14b	
%r15	%r15d %r15w		%r15b	

### Linux Yığıt (Stack) Yapısı



### Özel Hizalama Durumları (Intel IA32)

- 1 byte: `char`, ...  
*sınırlandırma yok*
- 2 bytes: `short`, ...  
*en düşük bit adresi  $0_2$*
- 4 bytes: `int`, `float`, `char *`, ...  
*en düşük 2 bit adresi  $00_2$*
- 8 bytes: `double`, ...  
Windows: *en düşük 3 bit adresi  $000_2$*   
Linux: *en düşük 2 bit adresi  $00_2$*
- 12 bytes: `long double`  
Windows & Linux:  
*en düşük 2 bit adresi  $00_2$*

C Veri Tipi	IA-32	X86-64
<code>char</code>	1	1
<code>short</code>	2	2
<code>int</code>	4	4
<code>long</code>	4	8
<code>long long</code>	8	8
<code>float</code>	4	4
<code>double</code>	8	8
<code>long double</code>	10/12	10/16
<code>pointer</code>	4	8

### Özel Hizalama Durumları (Intel x86-64)

- 1 byte: `char`, ...  
*sınırlandırma yok*
- 2 bytes: `short`, ...  
*en düşük bit adresi  $0_2$*
- 4 bytes: `int`, `float`, ...  
*en düşük 2 bit adresi  $00_2$*
- 8 bytes: `double`, `char *`, ...  
Windows & Linux:  
*en düşük 3 bit adresi  $000_2$*
- 16 bytes: `long double`  
Linux: *en düşük 3 bit adresi  $000_2$*

### Bayt Sıralama (Byte Ordering)

$0x100$  adresinde 4-baytlık değişken  
 $0x01234567$

#### Big Endian

*En anlamsız bayt en yüksek adreste*

$0x100$   $0x101$   $0x102$   $0x103$

01	23	45	67
----	----	----	----

#### Little Endian

*En anlamsız bayt en düşük adreste*

$0x100$   $0x101$   $0x102$   $0x103$

67	45	23	01
----	----	----	----

### Kayan noktalı sayı (floating point)

$$\text{Bias} = 2^{k-1} - 1$$

**Soru 1. (20 puan) Tamsayı gösterimleri.**

6-bit'lik bir bilgisayar üzerinde,

- *İşaretili tam sayılar (signed integers) için ikili tümler (2's complement) aritmetiği kullanılmaktadır.*
- *short tamsayılar 3-bit ile gösterilmektedir.*
- *Bir short açıkça int'e dönüştürülürken (cast edilirken) işaret genişletmesi (sign extension) kendiliğinden gerçekleşmektedir.*
- *int'ler üzerinde sağa kaydırma aritmetik kaydırma (arithmetic shift) işlemi ile gerçekleşmektedir.*

Bu varsayımlara göre aşağıdaki tanımları göz önünde bulundurarak altta verilen tablodaki boş kutucukları doldurunuz.

*NOT: “-” ile belirtilen bölümleri doldurmanıza gerek yoktur.*

```
short sa = -2;  
int b = 3*sa;  
int a = -16;  
short sb = (short) a;  
unsigned ua = a;
```

İfade	Tam sayı gösterimi	İkili gösterimi
<i>Sıfır</i>	0	000 000
(short) 0	0	000
-	29	011 101
-	-13	110 011
sa	-2	110
b	-6	111 010
sb	0	000
ua	48	110 000
a >> 2	-4	111 100
ua >> 2	12	001 100
b << 3	16	010 000
<i>Tmax</i>	31	011 111
<i>Tmax - Tmin</i>	-1	111 111

**Soru 2. (18 puan) Kayan noktalı sayı gösterimleri.**

Bu soruyu IEEE Standard 754 kayan noktalı sayı formatına göre oluşturulan 8-bit'lik bir kayan noktalı gösterimine göre cevaplayınız. Bu gösterimde,

- En anlamlı bit (the most significant bit) *işaret bit*'idir.
- İşaret bit'inin ardından gelen 3 bit kayan noktalı sayının *üstünü* (*exponent*) verir. Burada *üst için kaydırma değeri* (*exponent bias*) 3'tir.
- Geri kalan 4 bit ise *kesirli kısmı* (*fraction*) belirtir.
- Bu gösterimde ifade edilen sayılar,

$$V = (-1)^s \times M \times 2^E$$

şeklinde yazılabilen kayan noktalı sayılardır.

(*E*: kaydırılmış üst değeridir (*biased exponent*). *M*: *x* veya *x/y* şeklinde bir sayıya karşılık gelirken burada *x* bir tamsayı ve *y* ise 2'nin bir katıdır.)

Bu gösterim için belirlenen kurallar, *normalize olan sayı*, *normalize olmayan sayı*, *sonsuz* ve *NaN* (*Not a Number*) için IEEE Standard 754'e benzerdir. Buna göre aşağıda verilen tablodaki boş kutucukları doldurunuz.

*NOT*: “–” ile belirtilen bölümleri doldurmanıza gerek yoktur. Eğer bilgisayardaki gösterimde yuvarlama gerekiyor ise çifte-yuvarlama (round-to-even) yapmalısınız.

İfade	İkili değer	<i>M</i>	<i>E</i>	Değer
<i>Negatif sıfır</i>	1 000 0000	0	-2	-0.0
–	1 000 0101	5/16	-2	-5/64
<i>Normalize olan en küçük negatif sayı</i>	1 110 1111	31/16	3	-31/2
<i>Normalize olmayan en büyük pozitif sayı</i>	0 000 1111	15/16	-2	15/64
–	1 110 0101	21/16	3	-10.5
<i>Pozitif sonsuz</i>	0 111 0000	–	–	+∞

**Soru 3. (15 puan) Aritmetik optimizasyonu.**

Aşağıda bir grup C ve Assembly kodu verilmiştir.

```
foo1:                                int fun1(int x)
    pushl %ebp                        {
    movl %esp,%ebp                    {   return (x >> 31);
    movl 8(%ebp),%eax                 }
    sall $6,%eax                      }
    subl 8(%ebp),%eax
    movl %ebp,%esp                    int fun2(int x)
    popl %ebp                          {
    ret                                {   return 63 * x;
                                        }
}

foo2:                                int fun3(int x)
    pushl %ebp                        {
    movl %esp,%ebp                    {   return (x << 5) & 1;
    movl 8(%ebp),%ecx                 }
    movl %ecx,%eax
    sarl $31,%eax
    shrl $27,%eax
    addl %ecx,%eax                    int fun4(int x)
    sarl $5,%eax                       {
    popl %ebp                          {   return ((x + 31) - 27) /5;
    ret                                }
}

foo3:                                int fun5(int x)
    pushl %ebp                        {
    movl %esp,%ebp                    {   return x / 32;
    movl 8(%ebp),%eax                 }
    shrl $31,%eax
    movl %ebp,%esp                    int fun6(int x)
    popl %ebp                          {
    ret                                {   return (x < 0);
                                        }
}


```

Yukarıda sol tarafta verilen her bir Assembly programının yukarıda sağ tarafta verilen hangi C fonksiyonuna karşılık geldiğini açıklamasıyla birlikte belirtiniz:

- foo1 ≡ fun 2
- foo2 ≡ fun 5
- foo3 ≡ fun 6

**Soru 4. (24 puan) Assembly/C çevrimi.**

Aşağıda bir C fonksiyonu için derleyici tarafından üretilen Assembly kodu gösterilmektedir:

```
8048374 <foo>:
8048374:  push   %ebp
8048375:  mov    %esp,%ebp
8048377:  push   %ebx
8048378:  mov    0x8(%ebp),%eax
804837b:  mov    (%eax),%ebx
804837d:  mov    $0x0,%eax
8048382:  cmp    $0x3,%ebx
8048385:  jle    80483a4 <foo+0x30>
8048387:  mov    $0x0,%eax
804838c:  mov    $0x3,%ecx
8048391:  lea   (%eax,%ecx,2),%eax
8048394:  mov    %ebx,%edx
8048396:  sub   %edx,%eax
8048398:  sub   $0x1,%edx
804839b:  jne    8048396 <foo+0x22>
804839d:  add   $0x1,%ecx
80483a0:  cmp   %ebx,%ecx
80483a2:  jne    8048391 <foo+0x1d>
80483a4:  pop   %ebx
80483a5:  pop   %ebp
80483a6:  ret
```

Yukarıda verilen Assembly koduna göre aşağıdaki C kodunda yer alan boşlukları doldurunuz.

<pre>int foo (____ n) {      int a, i, j;      a = 0;     for(i ____; i ____; ____) {         a = a + ____;         for(j = ____ ;j ____; ____)             a = a - ____;     }      return a;  }</pre>	<pre>int foo (int *n) {      int a, i, j;      a = 0;     for(i=3; i&lt;(*n); i++)     {         a = a + 2*i;         for(j=(*n); j&gt;0; j--)             a = a - j;     }      return a;  }</pre>
---	---

**Soru 5. (24 puan) Yöntemler ve yığıt.**

Aşağıda özyinelemeli (recursive) bir fonksiyonun C kodu ve ilgili Assembly kodu verilmiştir:

<pre>int g(int n, int x, int y) {     if (n == 0)         return x;      return g(n-1, y, x+y); }</pre>	<pre>0x08048374 &lt;g+0&gt;: push    %ebp 0x08048375 &lt;g+1&gt;: mov     %esp,%ebp 0x08048377 &lt;g+3&gt;: sub    \$0x10,%esp 0x0804837a &lt;g+6&gt;: cmpl   \$0x0,0x8(%ebp) 0x0804837e &lt;g+10&gt;: je     0x80483a4 &lt;g+48&gt; 0x08048380 &lt;g+12&gt;: mov    0x10(%ebp),%eax 0x08048383 &lt;g+15&gt;: add   0xc(%ebp),%eax 0x08048386 &lt;g+18&gt;: mov   0x8(%ebp),%edx 0x08048389 &lt;g+21&gt;: sub   \$0x1,%edx 0x0804838c &lt;g+24&gt;: mov   %eax,0x8(%esp) 0x08048390 &lt;g+28&gt;: mov   0x10(%ebp),%eax 0x08048393 &lt;g+31&gt;: mov   %eax,0x4(%esp) 0x08048397 &lt;g+35&gt;: mov   %edx,(%esp) 0x0804839a &lt;g+38&gt;: call  0x8048374 &lt;g&gt; 0x0804839f &lt;g+43&gt;: mov   %eax,-0x4(%ebp) 0x080483a2 &lt;g+46&gt;: jmp   0x80483aa &lt;g+54&gt; 0x080483a4 &lt;g+48&gt;: mov   0xc(%ebp),%eax 0x080483a7 &lt;g+51&gt;: mov   %eax,-0x4(%ebp) 0x080483aa &lt;g+54&gt;: mov   -0x4(%ebp),%eax 0x080483ad &lt;g+57&gt;: leave 0x080483ae &lt;g+58&gt;: ret</pre>
---	--

Bir programın  $g(3, 0, 1)$  fonksiyon çağrısı gerçekleştirdiğini varsayınız. Bu çağrıdan önce  $\%esp$ 'nin değerinin  $0xffff1000$  olduğunu kabul ederek ( $0xffff1000$ , `call` komutunun çalıştırılmasından hemen önceki  $\%esp$ 'nin değeridir) aşağıdaki soruları yanıtlayınız.

**(a) (18 puan)**  $g(3, 0, 1)$  çağrısı sırasıyla şu fonksiyon çağrılarına neden olmaktadır:  $g(3, 0, 1)$ ,  $g(2, 1, 1)$ ,  $g(1, 1, 2)$  ve  $g(0, 2, 3)$ . Size sağlanmış kodu ve IA32 yığıt kurallarını göz önüne alarak bir sonraki sayfada verilmiş olan yığıt diyagramını  $g(1, 1, 2)$  için `leave` komutu çalıştırılmadan hemen öncesinde mevcut olan değerler ile doldurunuz.

Her yığıt alanı için (eğer mümkünse) içerdiği sayısal değerleri belirtiniz. Eğer bir alanın sayısal değerini hesaplamada size verilen bilginin yetersiz olduğunu düşünüyorsanız o alanın üstünü çarpıyla işaretleyebilirsiniz.

Bu soruyu yanıtlarken `leave` komutunun,

```
movl %ebp, %esp;
popl %ebp
```

ikili komut dizisine denk olduğunu unutmayınız.



x	0xffff100c
1	0xffff1008
0	0xffff1004
3	0xffff1000
x	0xffff0ffc
x	0xffff0ff8
x	0xffff0ff4
1	0xffff0ff0
1	0xffff0fec
2	0xffff0fe8
0x0804839f	0xffff0fe4
0xffff0ff8	0xffff0fe0
x	0xffff0fdc
2	0xffff0fd8
1	0xffff0fd4
1	0xffff0fd0
0x0804839f	0xffff0fcc
0xffff0fe0	0xffff0fc8
2	0xffff0fc4
3	0xffff0fc0
2	0xffff0fbc
0	0xffff0fb8
x	0xffff0fb4
x	0xffff0fb0
x	0xffff0fac

**(b) (6 puan)**  $g(1, 1, 2)$  için ret komutunun çalıştırılmasının hemen öncesinde `%esp` ve `%ebp`'nin değerleri nedir?

`%ebp: 0xffff0fe0`  
`%esp: 0xffff0fcc`

**Soru 6. (16 puan) Struct.**

Aşağıda Intel IA-32 bit Linux’da derlenmiş bir struct tanımı verilmiştir.

```
struct mystruct {
    long a;
    short b;
    int c;
    double d;
    char x[5];
    int* y;
    char z;
    float q;
}
```

(a) (8 puan) Altta verilen tablodaki her hücrenin büyüklüğünü 1 bayt kabul ederek yukarıda tanımı verilen mystruct veri tipinin hafızadaki yerleşimini ilgili hücreleri etiketleyerek gösteriniz. Bunu gerçekleştirirken bayt sıralama kurallarına uyunuz.

*NOT: Doldurma (padding) amacıyla kullanılan bölgeler varsa ilgili hücrelerin içeriğini karalayınız.*

a0	a1	a2	a3	b0	b1	=	=
c0	c1	c2	c3	d0	d1	d2	d3
d4	d5	d6	d7	x[0]	x[1]	x[2]	x[3]
x[4]	=	=	=	y0	y1	y2	y3
z	=	=	=	q0	q1	q2	q3

mystruct veri tipinden yaratılan ms adlı değişken için gdb’nin aşağıdaki bellek içeriği gösterdiğini varsayınız.

```
(gdb) x/40b &ms
0xffffcde0: 0xcb 0xce 0x66 0x45 0x41 0x29 0xd1 0x01
0xffffcde8: 0x7d 0x2a 0xff 0xff 0xbe 0xba 0xef 0xbe
0xffffcdf0: 0xc8 0x5b 0x70 0x6d 0x65 0xff 0xff 0xff
0xffffcdf8: 0x23 0xbb 0xdf 0x3e 0xf0 0xcd 0xff 0xff
0xffffce00: 0x43 0xf0 0x00 0x00 0xf4 0x7f 0x86 0x47
```

Buna yukarıdaki alanları etiketleyiniz ve aşağıdaki değerleri doldurunuz:

- ms.a = 0x4566cecb
- ms.b = 0x2941
- ms.c = 0xffff2a7d
- ms.d = 0x6d705bc8beefbabe
- ms.x = 0x65 , 0xff , 0xff , 0xff , 0x23
- \*(ms.y) = 0x6d705bc8
- ms.z = 0x43
- ms.q = 0x47867ff4

(b) (8 puan) mystruct veri tipinin içerdiği elemanlardan oluşan, bellekteki toplam büyüklüğü olabilecek en küçük değere sahip olan yeni bir struct tanımlayın.

```
struct mycompressedstruct
{
    double d;
    long a;
    int c;
    int* y;
    float q;
    short b;
    char x[5];
    char z;
}
```

mycompressed veri tipinin bellekteki yerleşimini ve bellekte ne kadarlık yer tuttuğunu gösteriniz.

d0	d1	d2	d3	d4	d5	d6	d7
a0	a1	a2	a3	c0	c1	c2	c3
y0	y1	y2	y3	q0	q1	q2	q3
b0	b1	x[0]	x[1]	x[2]	x[3]	x[4]	z

**Soru 7. (8 puan) Okuma ödevleri.**

- IEEE-754 kayan noktalı sayı formatında sıfır sayısının +0 ve -0 olarak iki farklı temsilinin olması ne gibi bir avantaj sağlamaktadır? 1-2 cümle ile açıklayınız.

Örneğin  $1 / (1/x)$  ifadesini ele alalım. Eğer  $x$ 'in değeri  $+\infty$  veya  $-\infty$  ise  $1/x$  ifadesi  $+0$  veya  $-0$ 'a eşit olur ve buna göre tüm ifadenin sonucu  $1/(+0)$  veya  $1/(-0)$ , yani  $+\infty$  veya  $-\infty$ 'a denk olur. Sıfırın tek bir temsili olsaydı bu ifadenin sonucu  $x$ 'in değerinin  $+\infty$  veya  $-\infty$  olduğu durumda daima  $+\infty$ 'a eşit olacaktı ve işaret bilgisini kaybedecektik.

- Hangi iki Bell Labs çalışanı hem Unix işletim sisteminin hem de C programlama dilinin geliştirilmesinde aktif rol almışlardır? Soyadlarını yazmanız yeterlidir.

**Dennis M. Ritchie and Ken Thompson**