Öğrenci Adı – Soyadı: _____
Öğrenci Numarası: _____

S1	*S2*	*S3*	*S4*	*S5*	*S6*	*S7*	*Toplam*

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`	*Src,Dest*	Dest = Dest + Src
`subl`	*Src,Dest*	Dest = Dest - Src
`imull`	*Src,Dest*	Dest = Dest * Src
`sall`	*Src,Dest*	Dest = Dest << Src
`sarl`	*Src,Dest*	Dest = Dest >> Src
`shrl`	*Src,Dest*	Dest = Dest >> Src
`xorl`	*Src,Dest*	Dest = Dest ^ Src
`andl`	*Src,Dest*	Dest = Dest & Src
`orl`	*Src,Dest*	Dest = Dest | Src
`incl`	*Src*	Dest = Dest + 1
`decl`	*Src*	Dest = Dest - 1
`negl`	*Src*	Dest = - Dest
`notl`	*Src*	Dest = ~ Dest

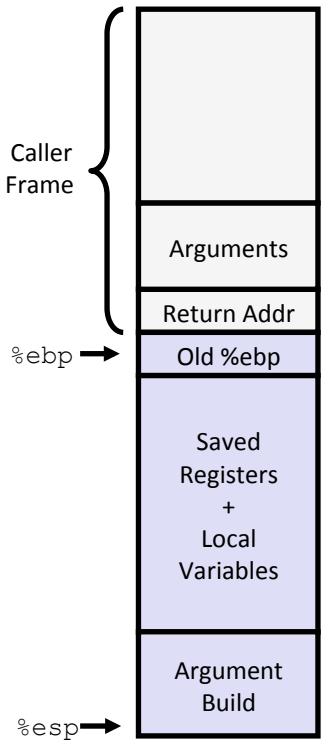
Bellek İşlemleri

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

Yazmaçlar (Registers)

63	31	15	8 7	0	
`%rax`	`%eax %ax`	`%ah`	`%al`		Return value
`%rbx`	`%ebx %bx`	`%bh`	`%bl`		Callee saved
`%rcx`	`%ecx %cx`	`%ch`	`%cl`		Argument #4
`%rdx`	`%edx %dx`	`%dh`	`%dl`		Argument #3
`%rsi`	`%esi %si`		`%sil`		Argument #2
`%rdi`	`%edi %di`		`%dil`		Argument #1
`%rbp`	`%ebp %bp`		`%bpl`		Callee saved
`%rsp`	`%esp %sp`		`%spl`		Stack Pointer
`%r8`	`%r8d %r8w`		`%r8b`		Argument #5
`%r9`	`%r9d %r9w`		`%r9b`		Argument #6
`%r10`	`%r10d %r10w`		`%r10b`		Reserved
`%r11`	`%r11d %r11w`		`%r11b`		Used for linking
`%r12`	`%r12d %r12w`		`%r12b`		Callee saved
`%r13`	`%r13d %r13w`		`%r13b`		Callee saved
`%r14`	`%r14d %r14w`		`%r14b`		Callee saved
`%r15`	`%r15d %r15w`		`%r15b`		Callee saved

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 *, ...
e*n düşük 2 bit adresi* 00_2

8 bytes: `double`, ...
Windows: e*n düşük 3 bit adresi* 000_2
Linux: e*n düşük 2 bit adresi* 00_2

12 bytes: `long double`
Windows & Linux:
e*n düşük 2 bit adresi* 00_2

C Veri Tipi	IA-32	X86-64
`char`	1	1
`short`	2	2
`int`	4	4
`long`	4	8
`long long`	8	8
`float`	4	4
`double`	8	8
`long double`	10/12	10/16
`pointer`	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`, ...
e*n düşük 2 bit adresi* 00_2

8 bytes: `double`, `char *`, ...
Windows & Linux:
e*n düşük 3 bit adresi* 000_2

16 bytes: `long double`
Linux: e*n düşük 3 bit adresi* 000_2

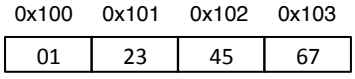
Kayan noktalı sayı (floating point)

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

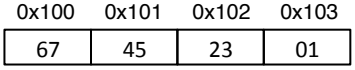
Bayt Sıralama (Byte Ordering)

`0x100` adresinde 4-bayt'lık değişken `0x01234567`

Big Endian
En anlamsız bayt en yüksek adreste



Little Endian
En anlamsız bayt en düşük adreste



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

6-bit'lik bir bilgisayar üzerinde,

- *İşaretli 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
Sıfır	0	000 000
(short) 0	0	000
–	29	
–		110 011
sa		
b		
sb		
ua		
a >> 2		
ua >> 2		
b << 3		
Tmax		
Tmax - Tmin		

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)* icin 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**	***M***	***E***	**Değer**
Negatif sıfır				-0.0
–	1 000 0101			
Normalize olan en küçük negatif sayı				
Normalize olmayan en büyük pozitif sayı				
–				-10.5
Pozitif sonsuz		–	–	+∞

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

Aşağıda bir grup C ve Assembly kodu verilmiştir.

```
foo1:
     pushl %ebp
     movl %esp,%ebp
     movl 8(%ebp),%eax
     sall $6,%eax
     subl 8(%ebp),%eax
     movl %ebp,%esp
     popl %ebp
     ret
```

```
int fun1(int x)
{
    return (x >> 31);
}

int fun2(int x)
{
    return 63 * x;
}
```

```
foo2:
     pushl   %ebp
     movl    %esp, %ebp
     movl    8(%ebp), %ecx
     movl    %ecx, %eax
     sarl    $31, %eax
     shrl    $27, %eax
     addl    %ecx, %eax
     sarl    $5, %eax
     popl    %ebp
     ret
```

```
int fun3(int x)
{
    return (x << 5) & 1;
}

int fun4(int x)
{
    return ((x + 31) – 27) /5;
}
```

```
foo3:
     pushl %ebp
     movl %esp,%ebp
     movl 8(%ebp),%eax
     shrl $31,%eax
     movl %ebp,%esp
     popl %ebp
     ret
```

```
int fun5(int x)
{
    return x / 32;
}

int fun6(int x)
{
    return (x < 0);
}
```

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

- `foo1 ≡ fun`

- `foo2 ≡ fun`

- `foo3 ≡ fun`

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.

```
int foo (____ n) {

  int a, i, j;

  a = 0;
  for(i ____; i ____; ____) {
      a = a + _____;
      for(j = ____ ;j ____; ____)
          a = a - ____;
      }

  return a;

}
```

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

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

```
int g(int n, int x, int y)
{
   if (n == 0)
     return x;

   return g(n-1, y, x+y);
}
```

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

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.

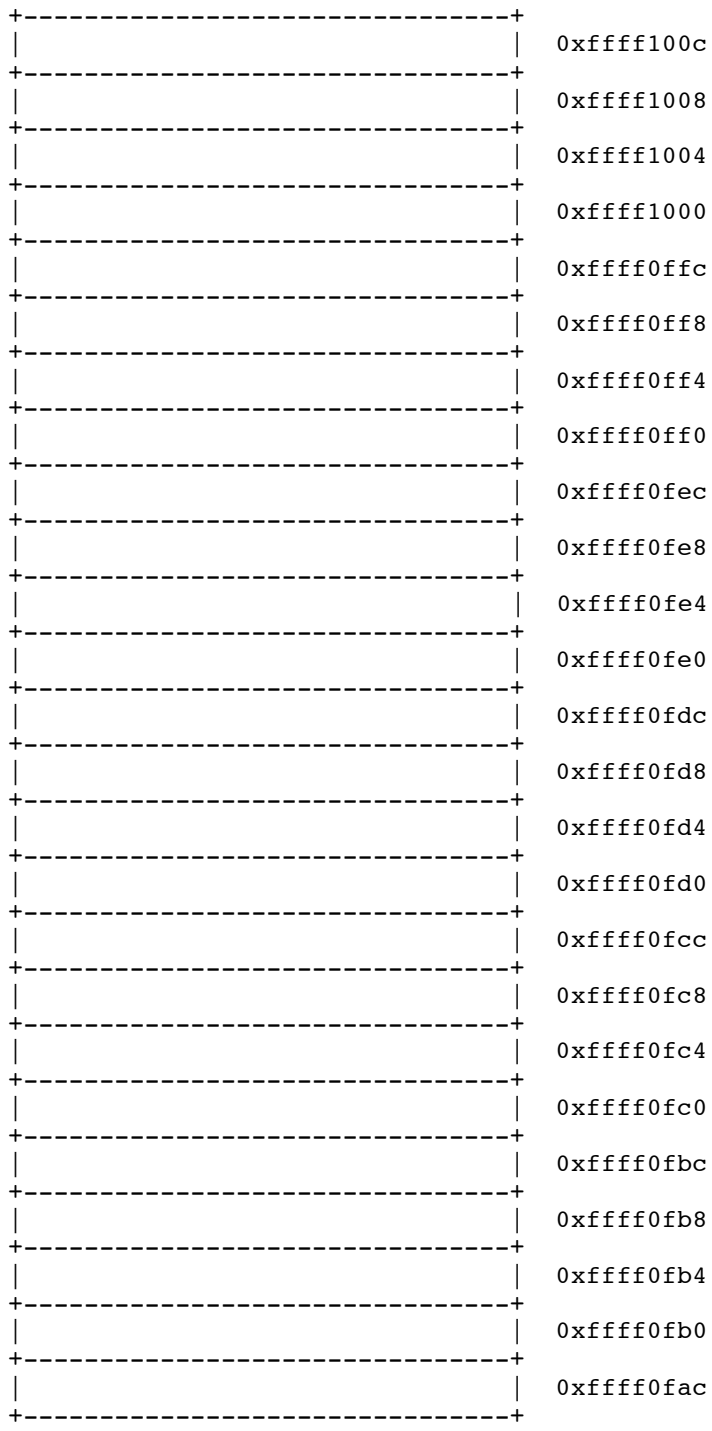
(a) (18 puan) `g(3,0,1)` çağrısı sırasıyla şu fonksiyon çağrılarına neden olmaktadır: `g(3,0,1)`, `g(2,1,1)`, `g(1,1,2)` ve `g(0,2,3)`. Size sağlanmış kodu ve IA32 yığıt kurallarını göz önüne alarak bir sonraki sayfada verilmiş olan yığıt diyagramını `g(1,1,2)` için `leave` komutu çalıştırılmadan hemen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



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

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 içini karalayınız.

`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
```

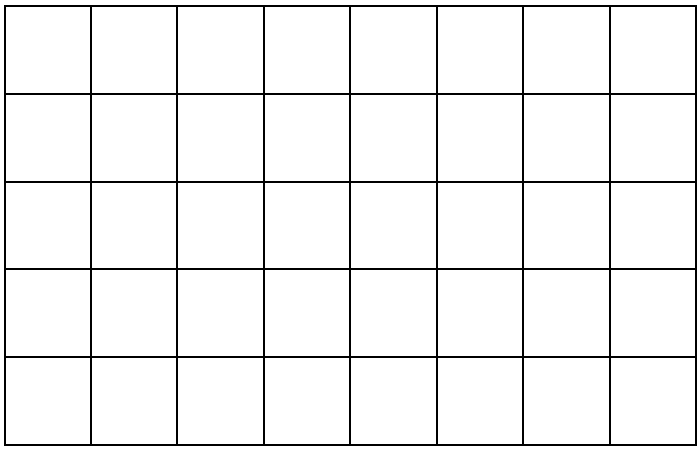
Buna yukarıdaki alanları etiketleyiniz ve aşağıdaki değerleri doldurunuz:

- `ms.a = 0x`
- `ms.b = 0x`
- `ms.c = 0x`
- `ms.d = 0x`
- `ms.x = 0x   ,0x   ,0x   ,0x   ,0x`
- `*(ms.y) = 0x`
- `ms.z = 0x`
- `ms.q = 0x`

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

```
struct mycompressedstruct
{









}
```

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



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.

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