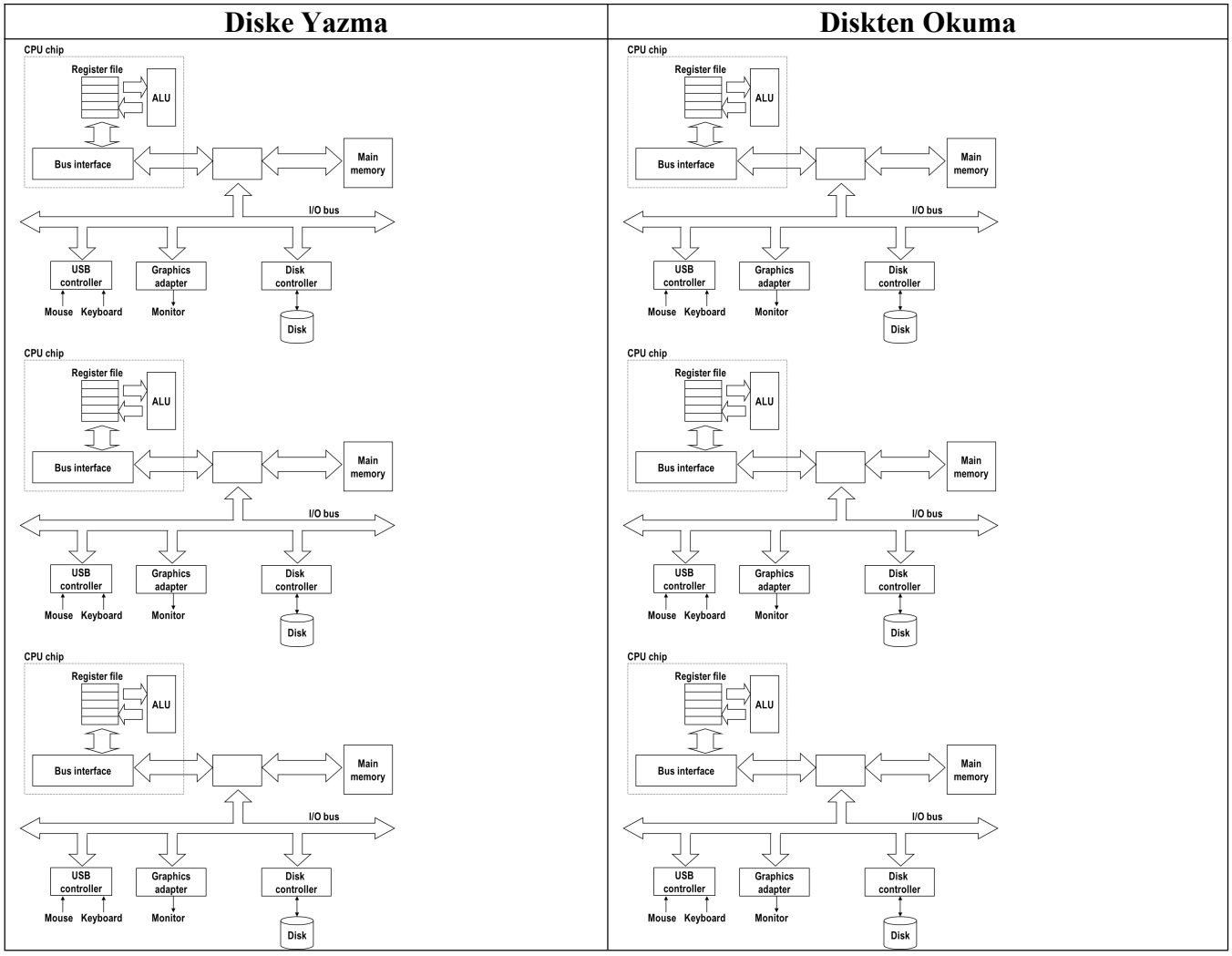
Hacettepe Üniversitesi
Bilgisayar Mühendisliği Bölümü
BBM341 Sistem Programlama
2. Ara sınav – 1 Aralık 2015

Öğrenci Adı: .....

Numarası: .....

Soru 1. (10 Puan) Doğrudan bellek erişim (DMA: *Direct Memory Access*) yöntemini kullanarak bir disk sektörünün erişimine ilişkin adımları aşağıdaki şekiller üzerinde gösteriniz.



Soru 2. (10 Puan) *Zamansal Yerellik* (*Temporal Locality*) kavramı aşağıdaki kod kesiminde hangi verilere erişim için söz konusudur, bir/iki cümle ile açıklayınız.

```
int sum_array_cols(int a[M][N])
{
    int i, j, sum = 0;
    for (i = 0; i < M; i++)
        for (j = 0; j < N; j++)
            sum += a[i][j];
    return sum;
}
```

.....
.....
.....
.....

Soru 3. (10 Puan) *Konumsal Yerellik* (*Spatial Locality*) kavramı aşağıdaki kod kesiminde hangi verilere erişim için söz konusudur, bir/iki cümle ile açıklayınız.

```
int sum_array_cols(int a[M][N])
{
    int i, j, sum = 0;
    for (i = 0; i < M; i++)
        for (j = 0; j < N; j++)
            sum += a[i][j];
    return sum;
}
```

Soru 4. (10 Puan) %99 "hit" oranıyla yapılan veri erişimleri %93 oranıyla yapılanlara göre kaç kat daha iyidir? Ön bellekten erişim zamanını 1 birim, ana bellekten erişim zamanını 50 birim alınız.

Soru 5. (20 Puan) Aşağıdaki C kaynak kodunda M ve N değişmezlerinin `#define` ile tanımlandığını varsayınız.

```
1     int mat1[M][N];
2     int mat2[N][M];
3
4     int sum_element(int i, int j) {
5           return mat1[i][j] + mat2[j][i];
6     }
```

Yukarıdaki C kaynak kodu derlendiğinde aşağıdaki liste elde edilmiştir. Buna göre tersine mühendislik yöntemini kullanarak M ve N değişmezlerinin değerini hesaplayınız.

```
   i at %ebp+8, j at %ebp+12
1     movl   8(%ebp), %ecx
2     movl   12(%ebp), %edx
3     leal   0(,%ecx,4), %eax
4     subl   %ecx, %eax
5     addl   %edx, %eax
6     leal   (%edx,%edx,8), %edx
7     addl   %ecx, %edx
8     movl   mat1(,%eax,4), %eax
9     addl   mat2(,%edx,4), %eax
```

M =

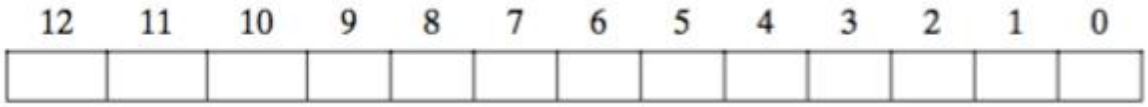
N =

Soru 6. (20 Puan) Aşağıda bir önbelleğin içeriği gösterilmiştir. Yazan tüm sayılar onaltılık sistemdedir. Her adres bir bayt uzunluğundaki bellek konumuna karşılık gelmektedir (4 baytlık değil). Adresler 13 bit uzunluğundadır. Ön bellekte 8 tane küme (S=8) vardır. Blok uzunluğu 4 bayttır (B=4). Her kümede iki satır vardır (E=2).

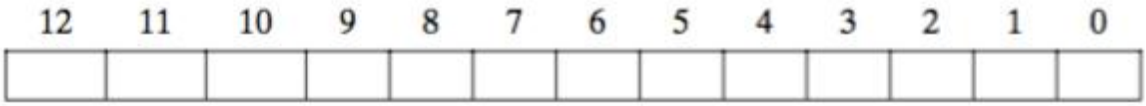
2-way set associative cache

	Line 0						Line 1					
Set index	Tag	Valid	Byte 0	Byte 1	Byte 2	Byte 3	Tag	Valid	Byte 0	Byte 1	Byte 2	Byte 3
0	09	1	86	30	3F	10	00	0	—	—	—	—
1	45	1	60	4F	E0	23	38	1	00	BC	0B	37
2	EB	0	—	—	—	—	0B	0	—	—	—	—
3	06	0	—	—	—	—	32	1	12	08	7B	AD
4	C7	1	06	78	07	C5	05	1	40	67	C2	3B
5	71	1	0B	DE	18	4B	6E	0	—	—	—	—
6	91	1	A0	B7	26	2D	F0	0	—	—	—	—
7	46	0	—	—	—	—	DE	1	12	C0	88	37

A. Aşağıdaki şekil her bit bir kutucuğa gelecek şekilde adres formatını göstermektedir. Kutucukları hangi alanların hangi amaçla kullanıldığını gösterecek şekilde işaretleyin. Kutucuklara ön bellek blok ofseti için CO, ön bellek küme indeksi için CI, ön bellek etiketi (tag) için CT yazınız.

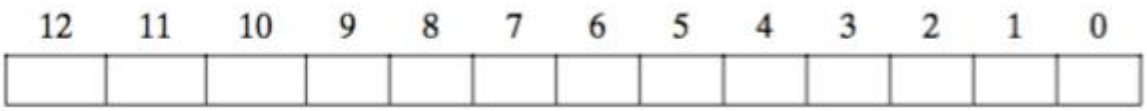


B. Bir program 0x1239 adresindeki bayta erişmek istiyor. Adresi ikilik sistemde yazınız.



Aşağıdaki alanları buna göre doldurunuz:
Önbellek blok içi adresi (offset): .....
Önbellek küme indeksi: .....
Önbellek etiketi (tag): .....
Önbellek hit/miss? .....
Önbellektekten dönen değer: .....

C. Bir program 0x1E1A adresindeki bayta erişmek istiyor. Adresi ikilik sistemde yazınız.



Aşağıdaki alanları buna göre doldurunuz:
Önbellek blok içi adresi (offset): .....
Önbellek küme indeksi: .....
Önbellek etiketi (tag): .....
Önbellek hit/miss? .....
Önbellektekten dönen değer: .....

Soru 7. (20 Puan) Aşağıdaki "`struct`" için bileşenlerin başlangıç adreslerini (offset) ve veri yapısının toplam uzunluğunu tabloya yazınız.

```
struct P4 { short w[4]; char *c[4] ; int i};
```

A. 32 bitlik mimari için:

w	c	i	Toplam uzunluk

B. 64 bitlik mimari için:

w	c	i	Toplam uzunluk