Organizasyon

1. 4-bit opcode, 12-bit adres kullanımı olan bir yapıda aşağıdaki komutlar da bulunmaktadır. Adres bilgisi, bellek adresini veya I/O port numarasını gösterir:

0001 : AC’yi bellekten yükle

0010 : AC’yi belleğe depola

0011 : AC’yi I/O’dan yükle

0101 : AC’ye bellek içeriğini topla

0111 : AC’yi I/O’ya depola

a) Şu 3 basamaklı işlemi yapacak program kodunu, 0300 bellek adresinden başlayarak yerleştiriniz (HEX formatta).

1. AC’yi 5 numaralı I/O birimden yükle
2. AC’ye 0940 bellek adresi içeriğini topla
3. AC’yi 6 numaralı I/O birimine depola

*Bellek Konumu İçerik*

*0300 3005*

*0301 5940*

*0302 7006*

b) Yazdığınız kodun işleyişini, adım adım ve PC, AC, IR, MAR, MBR içerikleri ile birlikte açıklayınız.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **PC** | **AC** | **IR** | **MAR** | **MBR** |  |
| 1 | 1 | **0300** | ? | ? | **0300** | ? | *PC deki programın başlangıç adresi MAR’ye yüklenir* |
| 2 | **0301** | ? | ? |  | **3005** | *MAR adres içeriği MBR’ye yüklenir, PC 1 artar* |
| 3 |  | ? | **3005** |  |  | *MBR içeriği IR’ye yüklenir* |
| 4 |  | ? |  | **0005** |  | *IR’deki komutun adres bölümü MAR’ye yüklenir* |
| 5 |  | ? |  |  | **(0005)** | *MAR’de gösterilen I/O port içeriği MBR’ye yüklenir* |
| 6 |  | **(0005)** |  |  |  | *MBR içeriği AC’ye yüklenir* |
| 2 | 1 |  |  |  | **0301** |  | *PC içeriği MAR’ye yüklenir* |
| 2 | **0302** |  |  |  | **5940** | *MAR adres içeriği MBR’ye yüklenir, PC 1 artar* |
| 3 |  |  | **5940** |  |  | *MBR içeriği IR’ye yüklenir* |
| 4 |  |  |  | **0940** |  | *IR’deki komutun adres bölümü MAR’ye yüklenir* |
| 5 |  |  |  |  | **(0940)** | *MAR’de gösterilen adres içeriği MBR’ye yüklenir* |
| 6 |  | **(AC)+(0940)** |  |  |  | *MBR içeriği AC’ye toplanır* |
| 3 | 1 |  |  |  | **0302** |  | *PC içeriği MAR’ye yüklenir* |
| 2 | **0303** |  |  |  | **7006** | *MAR adres içeriği MBR’ye yüklenir, PC 1 artar* |
| 3 |  |  | **7006** |  |  | *MBR içeriği IR’ye yüklenir* |
| 4 |  |  |  | **0006** |  | *IR’deki komutun adres bölümü MAR’ye yüklenir* |
| 5 |  |  |  |  | **(AC)** | *AC içeriği MBR’ye yüklenir* |
| 6 |  |  |  |  | (AC) | *MBR içeriği MAR’de gösterilen porta yüklenir* |

3. Bir “Set Associative” cache toplam 64 satırdan ve 4’er satırlık setlerden oluşmaktadır. Ana bellek ise, her biri 128 sözcük (word) içeren 4K bloğa ayrılmıştır.

1. Ana bellek adres formatını açıklayarak belirtiniz.

*1 Blok = 128 word = 27 →* ***w = 7 bit***

*64 satır cache, 1 set = 4 satır → k = 4 → set sayısı = 64/4 = 16 = 24 →* ***set = d = 4 bit***

*Blok sayısı = 4K = 4096 = 2S → s = 12 = TAG + set = TAG + 4 →* ***TAG = 8***

Tag = 8

Set = 4

Word = 7

1. Söz konusu cache organizasyon şemasını çizerek açıklayınız.



1. Kapasiteler ve blok (satır), sözcük sayıları aynı kalmak koşuluyla, cache organizasyonu “direct mapping” biçiminde olsaydı, 7BBBB adresindeki veri, cache alanının hangi satırının, hangi sözcüğüne, hangi etiket ile yerleşirdi?

*“direct mapping” organizasyonunda, bellek adresi “tag”, “line” ve “word” alanlarına ayrılır. “Word” aynı kalacaktır (7-bit). Cache alanı toplam 64 satırdan oluştuğuna göre, 64 = 26 → “line” = 6 bulunur. Toplam adres boyu 19-bit olduğuna göre, “tag” = 19-6-7 = 6 olacaktır.*

*19-bit kullanarak 7BBBB yazarsak ve bu 19 biti yukarıdaki gibi “tag”, “line” ve “word” alanlarına ayırırsak: 1 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 buluruz.*

Tag

Line

Word

*Her bölümü Hexadecimal olarak yazarsak: Tag = 3D; Line = 37; Word = 3B bulunur.*

4. 1024×1024×4 bellek çiplerinden yeteri kadar kullanarak 8Mbyte bellek organizasyonu şeması çiziniz. Çiziminizi ana hatları ile yapınız, tüm elemanları ve bağlantıları ayrı ayrı göstermeyiniz

Bellek yongaları 1024×1024 organizasyona sahip olduğuna göre, sütun ve satır seçimi söz konusudur. Buna göre 10-bit sütun, 10-bit de satır seçimi için kullanılacaktır (210 = 1024). Öte yandan her yonga 4-bit veri yoluna (data-bus) sahip olduğuna göre, bir byte oluşturmak için aynı adres ile 2 yonga seçilmelidir. Böylece her yonga çiftinden 1024×1024×8 bit, başka bir deyişle 1 MByte bellek oluşturulur. O halde, 8 MByte bellek için 8 çift yonga kullanılmalıdır. Bu 8 çiftin seçimi için ise 3-bit gerekecektir (23 = 8). Böylece toplam adres boyumuz 10+10+3 = 23-bit olmalıdır. Nitekim 223 = 8MByte olmaktadır. Çizimimizi buna göre yaparız:

10

10

3

sütun seç

Satır seç

1024×1024

×4

sütun seç

Satır seç

1024×1024

×4

sütun seç

Satır seç

1024×1024

×4

sütun seç

Satır seç

1024×1024

×4

sütun seç

Satır seç

1024×1024

×4

sütun seç

Satır seç

1024×1024

×4

Grup-1

Grup-2

Grup-8

Grup

seç

4

4

MBR

MAR

**2.** “4-way Set Associative” bir “cache” yapısında 64 adet set bulunmaktadır. Ana bellekteki toplam blok sayısı ise 65,536 olup, her blokda 8 adet sözcük (word) yer almaktadır.

1 sözcük = 1 byte olduğuna göre:

a) Bellek adres yapısını belirtiniz ve bu yapının bellek erişiminde nasıl kullanılacağını çizerek açıklayınız.

*cache’deki set sayısı = v = 64 olduğuna göre, set adres boyu =* ***d = 6-bit*** *olacaktır (26 = 64). Öte yandan her blokda 8 adet sözcük oluşu,* ***w = 3*** *olmasını gerektirir (23 = 8). Ana bellek kapasitesini bulmak için:*

*Ana bellekteki blok sayısı = Ana bellek büyüklüğü (byte) / Bir blok büyüklüğü (byte) →*

*65,536 = Ana bellek büyüklüğü / 8 → Ana bellek büyüklüğü = 8×65,536 = 524,288 = 512Kbyte bulunur. Buna göre ana bellek için toplam adres uzunluğu = s+w =* ***19-bit*** *olacaktır (219 = 512K). Bu durumda Tag = (s+w)-(w+d) = 19-(3+6) =* ***10-bit*** *olur. Adres yapısını buna göre çizersek:*

10 6 3

Tag Set Word

*Bu yapının bellek erişiminde nasıl kullanıldığı aşağıda gösterilmiştir. Buna göre, 6-bit set adresi ile cache’deki ilgili sete erişilir ve bu set içindeki 4 satırın (bloğun) 4 adet Tag’i adresin Tag bölümü ile ayrı ayrı karşılaştırılır. İsabet olursa, “word” ile belirlenen o blokdaki byte CPU’ya gönderilir. İsabet yoksa 19-bit adres ile (w = 000 yapılarak) ana belleğe erişilir ve erişilen satırdan başlayan 4 satırlık blok cache’e, ilgili setin herhangi bir satırına yerleştirilir (hangi satır olacağı “replacement algorithm” konusudur). Buradan da, istenen word CPU’ya gönderilir*

Byte = 0

Ana Bellek

⊗

Tag Set Byte

Blok – 0

Blok – 1

Blok – 2

Blok - 3

Set - 0

Blok – 4

Blok – 5

Blok – 6

Blok - 7

Set - 1

Blok – 252

Blok – 253

Blok – 254

Blok - 255

Set - 63

Cache

Compare

⊗

Hit in cache

10

6

3

Not hit in cache

To Cache

b) Belleğin 5B96D (Hex) adres içeriği cache’e taşındığında hangi setin hangi satırına oturur?

19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1

1 0 1 1 0 1 1 1 0 0 **1 0 1 1 0 1** 1 0 1

5 B 9 6 D

*Verilen adresin “set” adresini oluşturan 6 bit’i bold olarak gösterilmektedir. Bu da “101101” olup, ondalık 45 sayısına karşılık gelmektedir. O halde, söz konusu içerik set-45’e oturacaktır. Hangi satıra oturacağı ise bilinemez. Buna uygulanan “replacement algorithm” karar verir ve bu satır o setteki 4 satırdan herhangi birinin 5. sözcüğü olabilir (w = 101 = 5).*

c) Aynı temel yapının “direct mapping” uygulanan bir cache organizasyonu için nasıl bir bellek adres yapısını gerektireceğini açıklayarak gösteriniz (erişim çizimini yapmanıza gerek yoktur).

*Cache’deki set sayısı = v = 64 olduğunu biliyoruz. Her sette 4 satır bulunduğuna göre (4-way), cache’deki toplam satır sayısı = 4×64 = 256 olacaktır. Bu durumda söz konusu 256 satırı adresleyebilmek için r =* ***8-bit*** *(28 = 256) olmalıdır. Öte yandan,* ***w = 3-bit*** *ve toplam adres uzunluğu = s + w = 19-bit aynı kalacağına göre, Tag = (s + w) – (w + r) = 19 – 11 =* ***8-bit*** *olarak bulunur. Adres yapısını buna göre çizebiliriz:*

8 8 3

Tag Line Word

**3.** Aşağıdaki soruları yanıtlayınız:

a) Bir komut dönülünün belli başlı durumları ve simgeleri aşağıda gösterilmektedir. Buna göre, EXOR A,B komutunun komut dönülü süresince içinden geçeği durumları sırası ile, bu simgeleri kullanarak yazınız. (Verilen komut A ve B bellek adreslerinin içeriklerini EXOR işlemine sokar ve sonucu A adresine yazar)

iac : instruction address calculation

if : instruction fetch

iod : instruction operation decoding

oac : operand address calculation

of : operand fetch

do : data operation

os : operand store

EXOR *A,B komutu dönülünün belli başlı durumları, sırası ile:*

*iac – if – iod – oac – of – oac – of – do – oac – os olmaktadır. “iac” ile bir sonra işlenecek komutun adresi bulunur ardından “if” ile komut bellekten işlemciye okunur. “iod” ileyapılacak işlem çözümlendikten sonra 2 kez “oac” “of” ile A ve B operandları okunur. “do” ile EXOR işlemi yapıldıktan sonra, “oac” “os” ile sonuç A’ya yerleştirilir.*

1. Aşağıdaki yapının bellek haritasını bulunuz:

MEMW

°

°

°

MEMR

MEMR

MEMW

3×8 Decoder

A10

A0

D0 D7

CS R

2K × 8

EPROM

2

1

0

0

7

E

A15

°

3

D0 D3

A11

A0

CS R W

4K × 4

RAM

D4 D7

A11

A0

CS R W

4K × 4

RAM

A14

A13

A12

MEMR

A11

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 : 0000H

2K EPROM

0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 : 07FFH

0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 : 3000H

4K RAM

0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 : 3FFFH

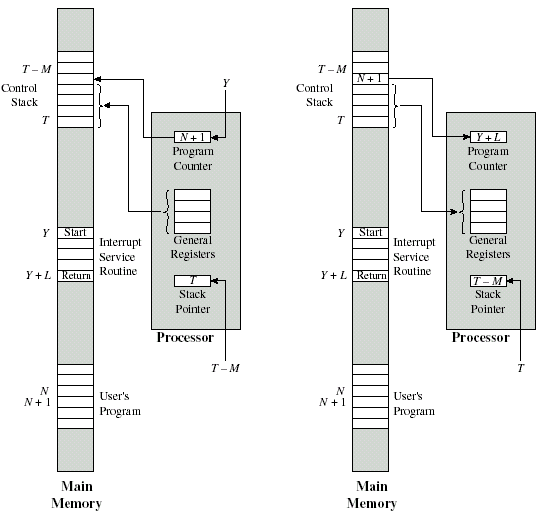
**4. a)**  Kesilme talebi (interrupt request) geldiği anda CPU’nun belleğin 548A adresindeki program komutunu işlemekte olduğunu, kesilme yordamının belleğin 2200 – 24FF adresleri arasında yer aldığını ve yığıt göstergesinin (stack pointer) 1189 içerdiğini varsayarak:

- Bellek tarafında yığıt içeriğinin

- CPU tarafında PC (Program Counter), genel yazmaçlar ve yığıt göstergesinin

Kesilme talebinden hemen önce, kesilme talebi işleme konduğu anda ve kesilme servis yordamından dönüş anında nasıl değiştiğini şematik bir anlatımla açıklayınız.

**NOT: Tüm değerler Hex olarak verilmiştir.**



1189

548B

1184

2200

548A

548B

2200

24FF

1184

1189

stack

24FF

1184

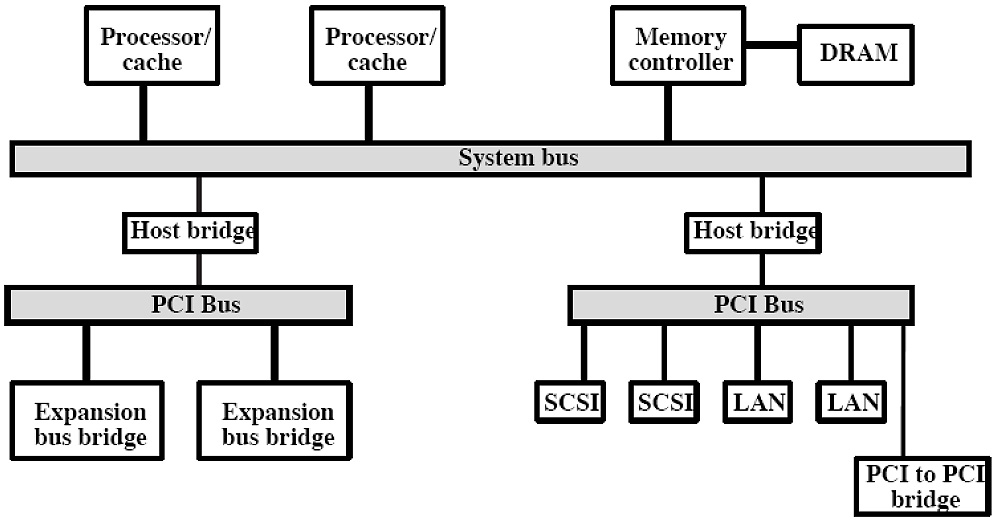
1189

548B

## Kesilmenin Başlangıcı

**Kesilmeden dönüş**

**b)** PCI bus kullanan tipik bir “server” sistem için bus organizasyon yapısını çiziniz ve bu yapının yüksek performans sağlamada nasıl işlev gördüğünü açıklayınız.



*“Multiprocessor” (çok işlemcili) yapılardada kullanılabilen bu “bus” yapısında, “system bus” sadece “processor/cache”, ana bellek ve “PCI bus” köprülerini desteklemekte, öteki tüm birimler, “system bus”a köprülerle bağlanmış yeteri kadar “PCI bus” üzerinde toplanmaktadır. “PCI bus”lar ile “system bus” bağlantılarını sağlayan köprüler, taşıdıkları tampon bellekler (buffer) sayesinde, farklı “bus” hızlarını destekleyebilmektedir. Ayrıca, bu yapıya, “expansion bus” köprüleri aracılığı ile, yeni genişleme “bus”ları eklenebilmektedir.*

2. a) Aşağıdaki yapının bağlı bulunduğu bilgisayarın adres uzayını (address space) belirtiniz ve bellek haritasını bulunuz. Bu yapıya sadece mevcut RAM yongalarından ekleyerek elde edilebilecek en büyük bellek kapasitesi ne olur? Nasıl sağlanır?

°

°

°

MEMR

MEMR

MEMW

3×8 Decoder

A10

A0

D0 D7

CS R

2K × 8

EPROM

2

1

0

0

7

E

A15

°

3

D0 D3

A11

A0

CS R W

4K × 4

RAM

D4 D7

A11

A0

CS R W

4K × 4

RAM

A14

A13

A12

MEMR

A11

MEMW

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 : 0000H

2K EPROM

0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 : 07FFH

0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 : 7000H

4K RAM

0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 : 7FFFH

3x8 decoder üzerinde 1,2,3,4,5,6 çıkışları boştur. Buralara da şekildeki gibi 4Kx4 RAM

Yonga çiftleri bağlanabilir. Böylece mevcut 4KB RAM kapasitesi 4x7=28KB olur. Bu

28KB RAM alanı, bellek haritasında 1000H – 7FFFH arasında yer alır.

1. Yapısında, ana bellek, cache, ve sanal bellek (virtual memory) olarak da kullanılan disk bulunan bir bilgisayarda, bir kelimenin cache’den okunması 15ns süre almaktadır. Aranan kelime cache’de bulunmuyor, ana bellekte bulunuyorsa, kelimenin önce cache’e yüklenmesi 50ns sürmekte ve ardından da cache’den okuma işlemi (yukarıdaki gibi) yapılmaktadır. Aranan kelime ana bellekte de yoksa, kelimenin diskten ana belleğe getirilmesi 10ms sürmekte, sonra, 50ns içinde cache’e yüklenmekte ve oradan, yukarıda açıklandığı gibi okunmaktadır. Arama işlemlerinin isabet oranları (hit ratio), cache için 0.85, ana bellek için ise 0.7 olduğuna göre, bir kelimenin ortalama okuma süresi (ns cinsinden) ne kadardır?

NOT: Hit ratio = 1, %100 isabet anlamına gelir.

|  |  |  |
| --- | --- | --- |
| **Aranan Kelimenin Yeri** | **Bulunma Olasılığı** | **Erişim Süresi (ns)** |
| Cache’de | 0.85 | 15 |
| Cache’de yok, ana bellekte var | (0.15)×(0.7) = 0.105 | 50 + 15 = 65 |
| Cache’de de yok, bellekte de yok | (0.15)×(0.3) = 0.045 | (10×106) + 50 + 15 = 10000065 |

*Ortalama erişim süresi için, 3 olası durumdaki erişim sürelerini, o durumların olasılıkları ile çarpar ve bulduğumuz sonuçları toplarsak:*

*[(0.85) × (15)] + [(0.105) × (65)] + [(0.045) × (10000065)] = 450022.5 ns buluruz*

3. 24-bit “address bus”, 16-bit “data bus” yapısı olan bir mikroişlemcide, 8 Kbyte, 4-way set-associative cache bellek bulunmaktadır. Cache’in her satırı (bloğu), “**byte-level” adreslenebilir** nitelikte 4 adet 16-bit kelimeden oluşmaktadır. 24-bit bellek adresinin hangi boylarda, kaç bölüme ayrılacağını ve bu yapının gerek cache, gerekse ana bellek adreslemede (“byte-level”) nasıl kullanılacağını, organizasyon şemasını çizerek açıklayınız Belleğin FF096A (Hex) adresinde kayıtlı bulunan bir verinin, cache yapısı içinde nerede bulunacağını bulunuz.

Aynı bus yapıları ve bellek büyüklükleri “direct mapping” ve “associative mapping” biçiminde organize edilseydi, 24-bit bellek adresi hangi bölümlerden oluşurdu?

*Blok büyüklüğü = 4 × 16 bit = 64 bit = 8× 8 bit = 8 byte*

*Blok içinde 8 byte adresleyebilmek için gerekli bit sayısı = log2(8) = 3 bit (offset).*

*Cache içindeki blok sayısı = 8 Kbyte / 8 byte = (8 × 1024) / 8 = 1024*

*Cache içindeki set sayısı = Cache içindeki blok sayısı / bir setteki blok sayısı*

*= 1024 / 4 = 256*

*256 adet seti adresleyebilmek için gerekli bit sayısı = log2(256) =* ***8***

*TAG için kullanılacak bit sayısı = 24 – (8 + 3) =* ***13***

*Bu veriler ışığında adres yapısını çizersek:*

Tag = 13

Set = 8

Byte = 3

*Verilen FF096A adresinin açılımı:* 1 1 1 1 1 1 1 1 0 0 0 0 1 0 0 1 0 1 1 0 1 0 1 0

*Görüldüğü gibi, set adresi (altı çizili 8 bit) = 00101101 = 2DH = 4510 olmaktadır. Buna göre verilen adres içeriği, cache’in 2D adresli setinin 4 bloğundan birinin, 2 numaralı byte’ında (byte = 010), 1111111100001 = 1FE1H etiketi ile bulunabilir*

*“Direct Mapping” yapılırsa, her blok (satır) yine 8 byte olacağına göre, cache alanı 8K/8 = 1024 satırdan oluşacaktır. 1024 satır, log2(1024) = 10 bit ile adreslenebilir. Yine Byte = 3 bit olacağına göre, Tag = 24-(10+3) = 11 bit olur. Buna göre adres yapısı şöyle olur:*

Tag = 11

Line = 10

Byte = 3

*“Associative” yapıda, “byte” ve “Tag” bulunacağından, adres yapısı aşağıdaki gibi olur:*

Tag = 21

Byte = 3

“Set Associative” yapının organizasyon şeması aşağıda gösterilmektedir:

Byte = 0

Ana Bellek

⊗

Tag (20) Blok (128)

Tag Set Byte

Blok – 0

Blok – 1

Blok – 2

Blok - 3

Set - 0

Blok – 4

Blok – 5

Blok – 6

Blok - 7

Set - 1

Blok – 1020

Blok – 1021

Blok – 1022

Blok - 1023

Set - 255

Cache

Compare

⊗

Hit in cache

13

8

3

Not hit in cache

To Cache

4. Aşağıda bir bellek kesiminin adres ve içerik dökümü gösterilmektedir. Bir varsayımsal (hypothetical) makinenin komut kümesi içinde ise, şu 3 komutun yer aldığıbilinmektedir:

HEX Kod Komut Açıklama

5XXX Mov M, AC XXX bellek adresi içeriğini AC’ye aktar

6XXX Mov AC, M AC içeriğini XXX bellek adresine aktar

7XXX Add M, AC XXX bellek adresi içeriğini AC’ye topla

Aşağıdaki tablonun boş gözlerini, belirtilen noktadan itibaren, her satır bir adıma karşılık gelecek ve tüm adımlar ilgili sütun başlıklarının içerik değişimini gösterecek biçimde doldurunuz. Bunu yaparken içeriğinin ilgisiz bulduğunuz gözler olursa, buralara X koyunuz.

NOT: “Cycle” sütununda, “Fetch” ya da “Execute” dönüllerinden hangisi olduğu belirtilecektir. Tablodaki satır sayısı rastgele seçilmiş olup, yazmanız gereken adım sayısını göstermeyebilir.

Bellek:

|  |  |
| --- | --- |
| Adres | İçerik |
| 4B8 | 57F8 |
| 4B9 | 77F9 |
| 4BA | 67FA |
| : | : |
| 7F8 | 5200 |
| 7F9 | 2187 |
| 7FA | 3DFF |

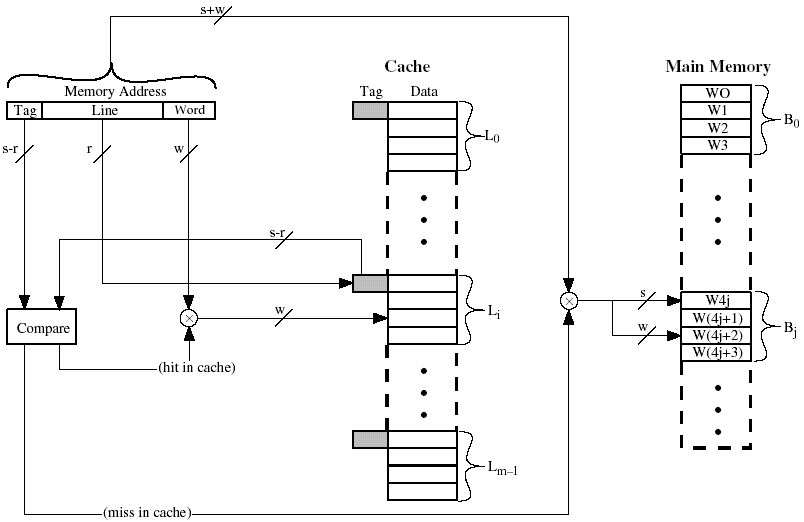
|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Cycle | 7F8 | 7F9 | 7FA | PC | AC | IR | MAR | MBR |
| Fetch | 5200 | 2187 | 3DFF | 4B8 | X | 57F8 | 4B8 | 57F8 |
| Execute | 5200 | 2187 | 3DFF | 4B9 | 5200 | 57F8 | 7F8 | 5200 |
| Fetch | 5200 | 2187 | 3DFF | 4B9 | 5200 | 77F9 | 4B9 | 77F9 |
| Execute | 5200 | 2187 | 3DFF | 4BA | 7387 | 77F9 | 7F9 | 2187 |
| Fetch | 5200 | 2187 | 3DFF | 4BA | 7387 | 67FA | 4BA | 67FA |
| Execute | 5200 | 2187 | 7387 | 4BB | 7387 | 67FA | 7FA | 7387 |

1. 64 KBytes ana bellekli bir yapıda, 2 KBytes cache bellek kullanılması ve her blokda 16 word bulunması istenmektedir. Bellek organizasyonu, 1 Byte = 1 wordolaraktasarlanmıştır**.**
   1. Bu yapıda, en az kaç bit bellek adresi kullanılması gerektiğini belirtiniz.

*log2 (65,536) = 16, En az 16 bit adres gerekir. (64K = 216)*

* 1. “Direct Mapping” yöntemi için, söz konusu bellek adresinin hangi boylarda, kaç bölüme ayrılacağını ve bu yapının gerek cache, gerekse ana bellek adreslemede nasıl kullanılacağını, organizasyon şemasını çizerek açıklayınız.

*Her blokda 16 word olacağına ve toplam cache alanı da 2048 Bytes (2 KBytes) olduğuna göre, cache alanı toplam 2048/16 = 128 blokdan (satırdan) oluşacaktır. 128 bloğu adresleyebilmek için, log2(128) = 7 bit gerekir (27 = 128). O halde, r = 7 olmalıdır. Öte yandan, her blokdaki 16 word’ü adresleyebilmek için de log2(16) = 4 bit gerekir (w = 4). Bu durumda, toplam adres boyu 16 bit olduğuna göre (s+w = 16), Tag için ayrılacak bit sayısı: (s + w) – (w + r) = 16 – (4 +7) = 5 olacaktır. bellekten bir word okunmak istendiğinde, önce adresin r bit bölümü (line) ile cache satırına erişim yapılır ve adresin ilk 5 biti (tag) o satırın etiketi (tag) ile karşılaştırılır. Tag’lar aynı ise (hit), adresin son 4 biti (w) kullanılarak istenen word okunur. Tag’lar aynı değilse, önce s+w bit adres ile (blok başına ulaşmak için w=0000 kullanılır) bellekten bir blok cache’a kopyalanır ve adresin son 4 biti kullanılarak istenen word okunur.*



:

:

:

:

:

W0

W15

W15

W127

* 1. Cache alanının kaç satırdan (kaç bloktan) oluşacağını ve ana bellekteki toplam blok sayısını belirtiniz. Ana bellekte ilk bloğun “Blok-0” olarak numaralandığını ve cache satırlarının “Satır-0”dan başlandığını varsayarak, bellekteki “Blok-132”nin hangi cache satırına haritalandığını bulunuz.

*(b) bölümünde açıklandığı gibi, cache alanı 128 blokdan (satırdan) oluşmaktadır. Benzer düşünüşle, ana bellek 64Kbytes ve her blok da 16 word (bytes) olduğuna göre, ana bellekdeki toplam blok sayısı: 65,536/16 = 4096 olacaktır (64K/16 = 4K).*

*i = j mod m (i = cache satır no., j = ana bellek blok no., m = cache’daki toplam satır sayısı) olduğuna göre, bellekteki Blok-132, 4. cache satırına yerleşecektir (132/128 işleminin artanı).*

* 1. Aynı temel yapı, “2-way Set Associative Mapping” yöntemi için organize edilse idi, bellek adresi hangi boylarda, kaç bölüme ayrılırdı? Bu durumda, bellekdeki “Blok-132”, cache alanında hangi sete (“Set-0”dan başladığını varsayarak) haritalanırdı?

*“2-way Set Associative Mapping” yönteminde, her set içinde 2 blok bulunacaktır. Bir blokdaki word sayısı ise, problem tanımı uyarınca yine 16 olacaktır. Bloklardaki word sayısı aynı kaldığı için de, cache alanındaki toplam blok sayısı (128) ve ana bellekteki toplam blok sayısı da (4096), direct-mapping ile aynı kalacaktır. Her set 2 blok içereceğine göre, cache alanı 128/2 = 64 set’ten oluşacaktır. 64 seti adreslemek için, log2(64) = 6 bit adres kullanmak gerekir. Her blokdaki 16 adet word ise, yine 4 bit ile adreslenecektir. Buna göre Tag için kalan bit sayısı, bellek adresi boyundan (16) bu iki sayı çıkartılarak kolayca bulunur: 16 – (6+4) = 6. Sonuç olarak, “2-way Set Associative Mapping” yönteminde bellek adresi 6-bit Tag, 6-bit Set ve 4-bit word alanlarından oluşacaktır. Bu kez, i = j mod v (i = cache set no., j = ana bellek blok no., v = toplam set sayısı) olacağından, Blok-132, 4. sete yerleşecektir (132/64 işleminin artanı).*

5. Aşağıdaki yapının bellek haritasını bulunuz (HEX olarak ifade ediniz)

A9

A0

A10

A0

MEMR

D0 D7

CS R

2K × 8

EPROM

A10

A15

A14

E

A13

A12

A11

3×8 Decoder

7

0

4

MEMR

D4 D7

MEMW

CS R W

1K × 4

RAM

MEMR

MEMW

D0 D3

1K × 4

RAM

A9

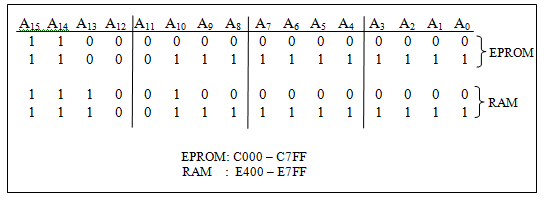
A0

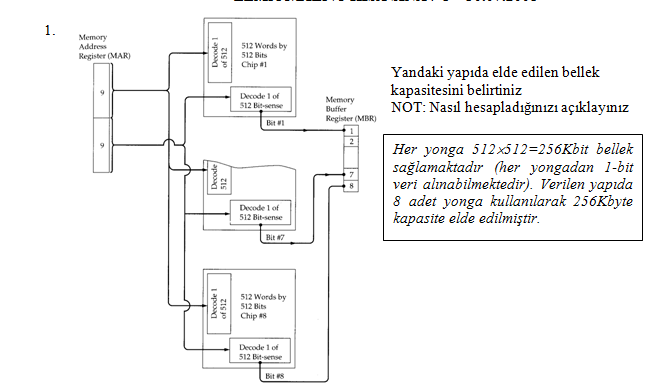
CS R W

2

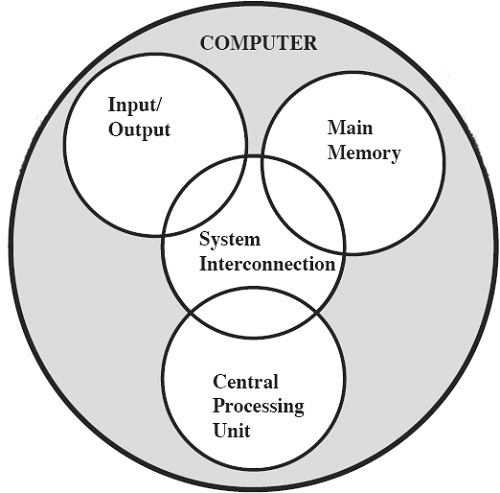
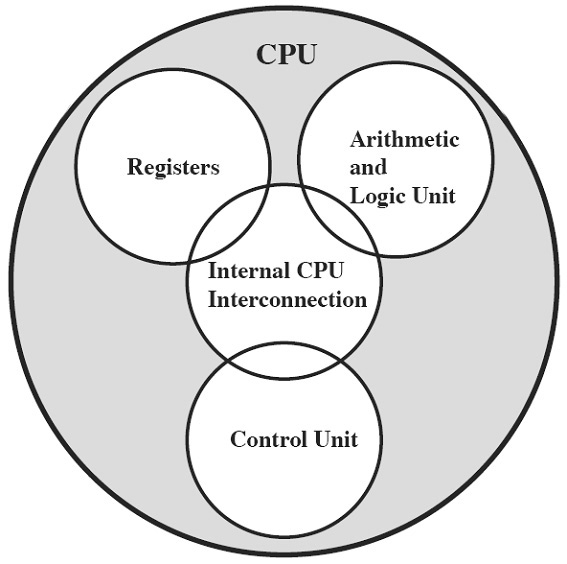
1

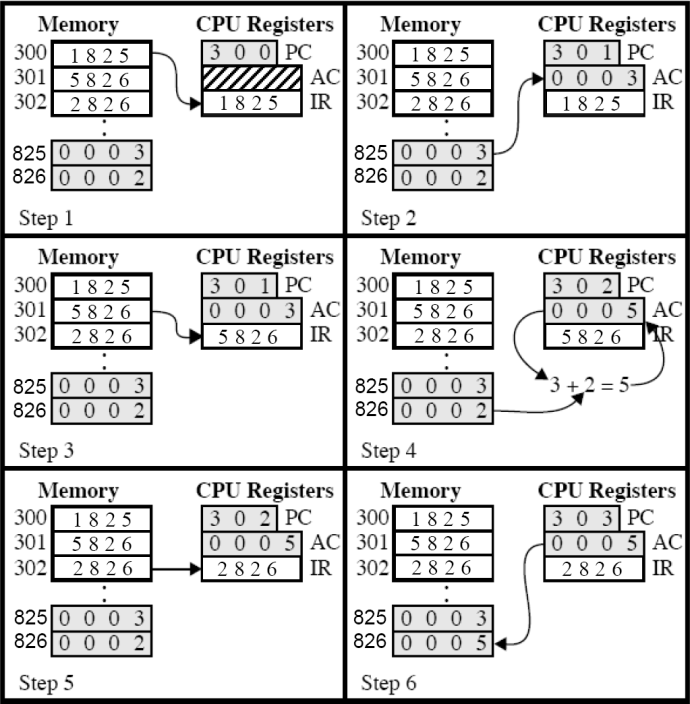
0





Aşağıdaki gri çemberin “bilgisayar” ve “merkezi işlem birimi” olması durumlarını ele alarak, içerideki çemberlere en uygun adları, her iki durum için de ayrı ayrı yazınız.





4-bit opcode, 12-bit adres kullanımı olan bir yapıda, aşağıda hexadecimal olarak gösterilen komutlar bulunmaktadır. Adres bilgisi, bellek adresini veya I/O port numarasını gösterir:

0001 : AC’yi bellekten yükle

0010 : AC’yi belleğe depola

0011 : AC’yi I/O’dan yükle

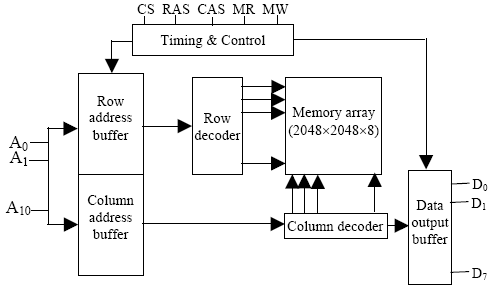
0101 : AC’ye bellek içeriğini topla

0111 : AC’yi I/O’ya depola

Buna göre, soldaki şekilde, içi boş olan gözlerde bulunması gereken içerikleri hexadecimal olarak yazınız (şekildeki değerler hexadecimal’dir).

* 1. Kesilme (interrupt) sürecinin nasıl gerçekleştiğini, 6-7 temel eylem olarak sıralayınız

1. *İlgili birim, CPU’ya kesilme istemi (sinyali) yollar.*
2. *CPU, her komutun son dönülünde kesilme istemi olup olmadığını denetler*
3. *Kesilme istemi varsa, o anda işlediği komutu tamamlar, kritik içerikleri saklar*
4. *PC içeriğine, kesilme servis yordamının başlangıç adresini koyar*
5. *Kesilme servis yordamını işletir*
6. *Kesilme servis yordamı bittiğinde, sakladığı kritik içerikleri geri koyar*
7. *Kesilmeden önceki programa kaldığı yerden devam eder*
   1. Adres ve veri için ayrılmış toplam pin sayısı 19 olan **bir adet** bellek yongası ile, 8-bit veri yolu (data bus) yapısında kullanılmak üzere, 32 Mbit kapasite elde edebilmek için, söz konusu yonga nasıl organize edilmelidir?

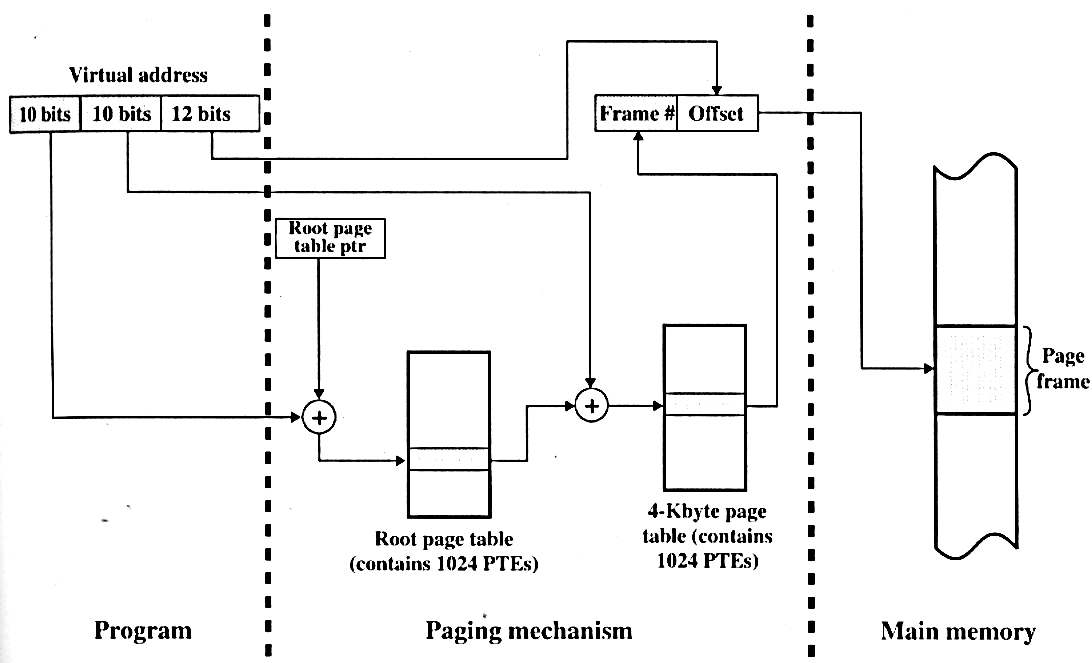


Veri yolu (data bus) 8-bit (1 byte) ya da 23 bit olduğuna göre, adres yolu (address bus) 25-3 = 22 bit olacaktır. Belleği “array” olarak organize edersek, bunun yarısı kadar pin kullanarak, 11 bit adresi “Row Address Select” (RAS) ve “Column Address Select” (CAS) aracılığı ile önce satır adresini, sonra da sütun adresini ilgili “buffer” alanlarına taşımakta kullanabiliriz. Bu da bize 211×211×23 = 2048×2048×8 = 32 Mbit verir. Bu durum yukarıdaki şekilde gösterilmektedir. “Column Decoder” tarafından seçilen her sütunda 8-bit birden seçilir.32bit= 25×220 = 225 demektir.

32-bit adres kullanan, byte seviyesinde adresleme yapılabilen bir sanal adres ortamı (virtual address space), her biri 8 Kbyte olan sayfalara ayrılmıştır. Sayfa tablosundaki (page table) her kayıt 4-byte uzunluktadır. Buna göre:

* + - Sanal bellek alanının kaç sayfadan oluştuğunu bulunuz
    - Gereksinim duyulan sayfa tablosunun büyüklüğünü bulunuz
    - Bellekte sürekli duracak bölümü (root table) 4 Kbyte olan 2 seviyeli bir sayfalama düzeni için kurgulayacağınız yapıyı ve bu yapıda 32-bit sanal adresi nasıl kullanacağınızı çizerek açıklayınız.

*Her bir sayfa içeriği, 8 Kbyte = 213 bit ile adreslenebilir. Sanal adres alanı 32 bit ile adreslendiğine göre, sanal bellek alanı, 232 / 213 =* ***219*** *sayfadan oluşmaktadır. Bu durumda, sayfa tablosunda (page table) toplam 219 kayıt (PTE) bulunacaktır. Sayfa tablosundaki her kayıt 4 byte olduğuna göre, sayfa tablosunun büyüklüğü 219 × 22 = 221 =* ***2 Mbyte*** *olur. 2 seviyeli sayfalama kurgusunu buna göre çizersek:*



**9**

**13**

Root page table

(1024 × 4 = 4Kbyte)

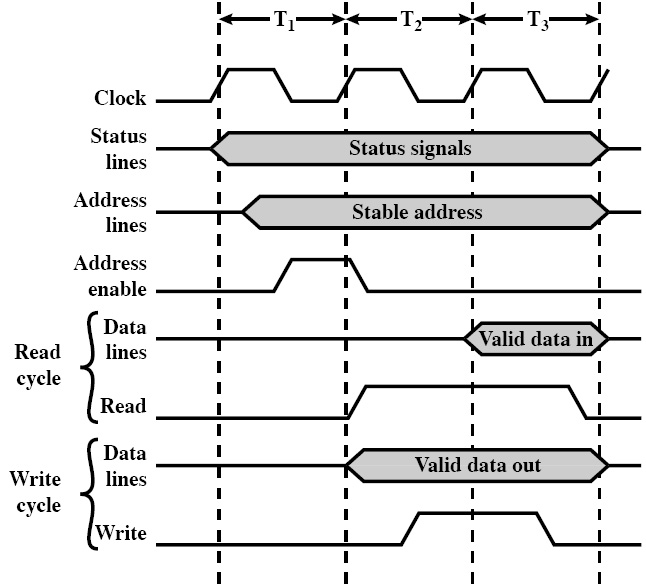
Page table

(512 × 4 = 2 Kbyte)

1024 adet

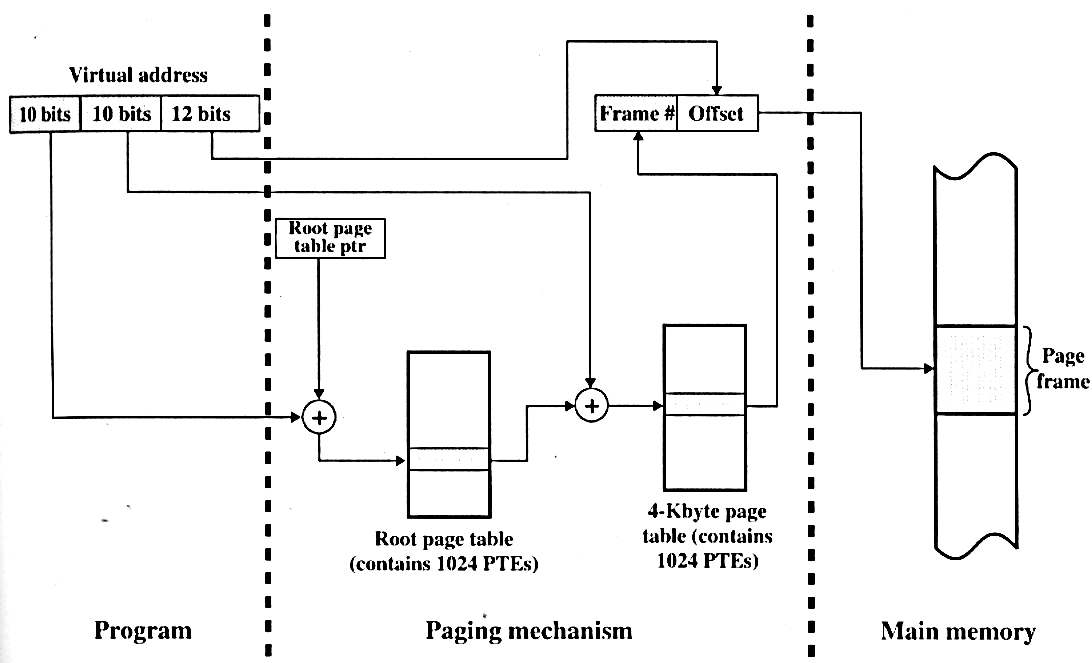
(toplam 2 MB)

* 1. 3 saat dönüllük bir süreç içinde, eşzamanlı “oku” (read) ve “yaz” (write) dönülünü, “Adres”, “Veri”, “Kontrol” (read-write) sinyal değişimlerini alt alta çizerek açıklayınız.



Bellekte sürekli duracak “root page table” 2Kbyte olan 2-seviyeli bir sayfalama düzeni (2-level paging) için kurgulayacağınız yapıyı ve 30-bit sanal adresi nasıl kullanacağınızı çizerek açıklayınız.

Bir sayfa tablosu 2 Kbyte = 211olduğundan adresin ofset bölümü 11-bit olacaktır. Öte yandan, bellekte sürekli duracak kök sayfa tablosunun (root page table) 2Kbyte büyüklüğünde olması için, (her kayıt 2-byte) 1024 satırdan oluşması gerekir. Bu da kök tablo satırlarının 10-bit ile adreslenmesi demektir. İkinci seviye kullanıcı sayfa tablosunun 1Mbyte olacağı yukarıda belirtilmişti. Kök tablodan toplam 1024 ayrı kullanıcı tablosu adreslenebileceğine göre, bu büyüklük için her tablo 1Kbyte olmalıdır. Her satır 2 byte olduğuna göre de, 512 satır bulunmalıdır (512=29). Bu da kullanıcı sayfa tablosu satırlarının 9-bit ile adresleneceği anlamına gelir. Esasen aynı değer sanal adres uzunluğu olan 30 dan, ofset boyu (11) ile kök tablo satır adresi uzunluğunun (10) çıkartılması ile de bulunabilir.



**9**

**11**

Root page table

(1024 × 2 = 2Kbyte)

Page table

(512 × 2 = 1 Kbyte)

1024 adet

(toplam 1 MB)

1. Elinizde aşağıdaki 3×8 decoder’den 1 adet ve bellek yongalarından da yeteri kadar bulunduğunu varsayarak, 0000 – 7FFF (Hex) adresleri arasında yer alacak 32 Kbyte RAM yapısına uygun bağlantı şemasını çiziniz.

2

1

0

0

7

E

3×8 Decoder

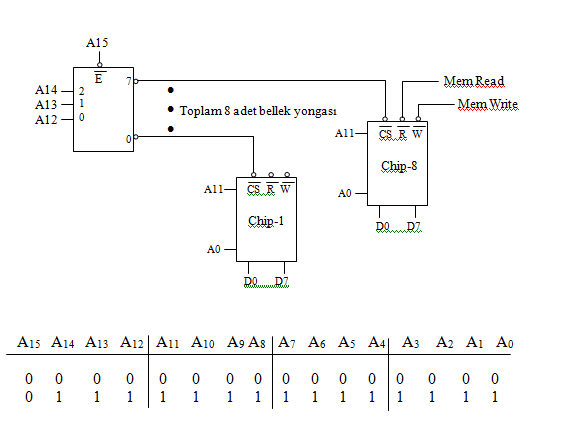
D0 D7

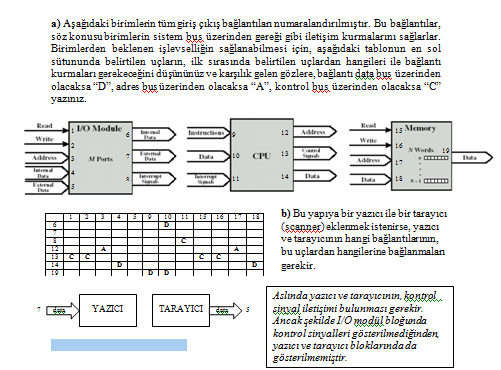
A11

A0

CS R W

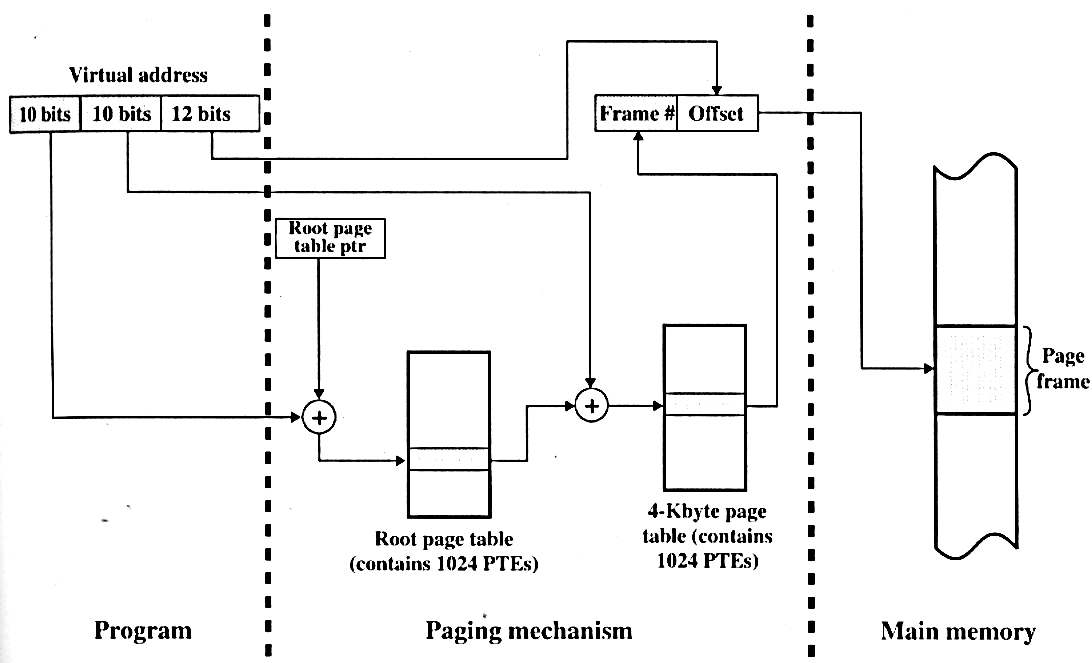
4K × 8 RAM





* 1. 32-bit ile “byte-level” adreslenen bir sanal adres ortamı (virtual address space), her biri 8 Kbyte olan sayfalara ayrılmıştır. Sayfa tablosundaki (page table) her kayıt 4-byte uzunluktadır. Buna göre:
     + Sanal bellek alanının kaç sayfadan olştuğunu bulunuz
     + Gereksinim duyulan sayfa tablosunun büyüklüğünü bulunuz
     + 2 seviyeli bir sayfalama düzeni (2-level paging) için kurgulayacağınız yapıyı ve 32-bit sanal adresi nasıl kullanacağınızı çizerek açıklayınız.

*Her bir sayfa içeriği, 8 Kbyte = 213 bit ile adreslenebilir. Sanal adres alanı 32 bit ile adreslendiğine göre, sanal bellek alanı, 232 / 213 =* ***219*** *sayfadan oluşmaktadır. Bu durumda, sayfa tablosunda (page table) toplam 219 kayıt (PTE) bulunacaktır. Sayfa tablosundaki her kayıt 4 byte olduğuna göre, sayfa tablosunun büyüklüğü 219 × 22 = 221 =* ***2 Mbyte*** *olur.2 seviyeli sayfalama kurgusunu buna göre çizersek:*



**9**

**13**

Root page table

(1024 × 4 = 4Kbyte)

Page table

(512 × 4 = 2 Kbyte)

1024 adet

(toplam 2 MB)

0000H – 07FFH Monitor Program (EPROM’da duracak)2000H – 23FFH Kullanıcı belleği

