- MİKROİŞLEMCİLER DERSİ PROJE RAPORU -

-8086 YILAN OYUNU-

SAKARYA
UYGULAMALI BİLİMLER
ÜNİVERSİTESİ

Ders Koordinatörü:

Prof. Dr. Halit Öztekin

HAZIRLAYANLAR:

FATMA YAŞAR - 23010903055

SÜMEYYE GÜL - 23010903049

CEREN ÖZKAN - 24010903131

ÖMER GÖKTÜRK - 23010903095

YAVUZ SELİM ÖZYURT - 23010903053

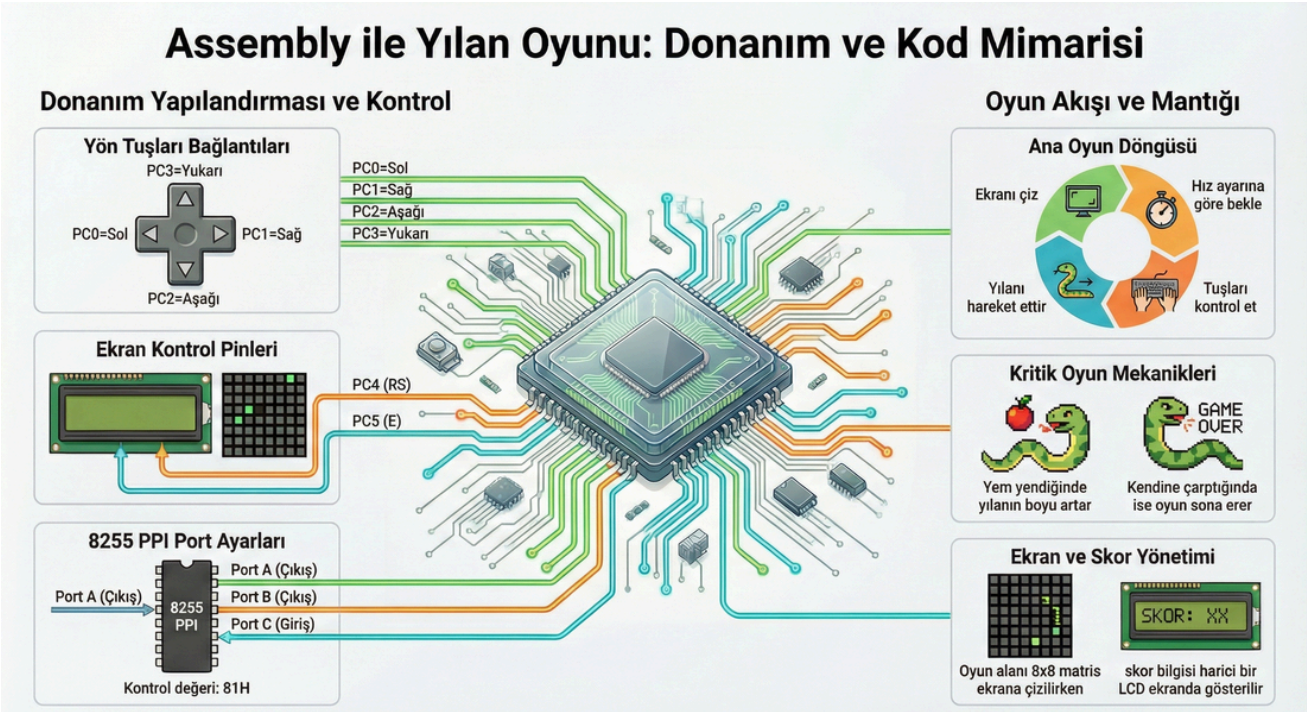
1. Proje Amacı

Bu projenin amacı, Intel 8086 mikroişlemcisi kullanılarak kullanıcı etkileşimli bir Yılan Oyunu tasarlamak, yılan hareketlerini yön butonları ile kontrol etmek ve skor bilgisini gerçek zamanlı olarak LCD ekran üzerinden gösterebilmektir.

2. Proje Konusu

Proje, 8x8 Dot Matrix paneli üzerinde klasik Yılan Oyunu'nun görselleştirilmesini kapsamaktadır. Yılanın hareketleri, kullanıcı tarafından yön butonları aracılığıyla kontrol edilirken, oyunda beliren yemler tamamen dinamik ve rastgele konumlandırılmaktadır; böylece oyun akışı her seferinde farklı bir deneyim sunar. Kullanıcı yemi yedikçe, oyunun ilerleyişine bağlı olarak skor güncellenir ve bu bilgi eş zamanlı olarak LM016L 16x2 karakter LCD ekran üzerinde görüntülenir. Sistem, oyun mantığını gerçek donanım platformu üzerinde çalıştıracak şekilde tasarlanmış olup, 8086 mikroişlemcinin kontrolünde, 8255 Programlanabilir Giriş/Çıkış arayüzü ve 74LS373 latch ile veri yollarının yönetilmesi sağlanmaktadır. Bu yapı sayesinde oyun, hem kullanıcı etkileşimli bir deneyim sunmakta hem de mikroişlemci tabanlı sistemlerin giriş/çıkış yönetimi, paralel veri iletimi ve donanım-yazılım entegrasyonu gibi temel prensiplerini ortaya koymaktadır.

Proje kapsamında, **Versiyon 1** simülasyon kararlılığı ve yazılım doğrulaması için temel referans mimari olarak baz alınmış; **Versiyon 2** ise sisteme fiziksel RAM ve ROM birimlerinin dahil edildiği, gerçek dünya mikrobilgisayar hiyerarşisini temsil eden ileri düzey donanım tasarımı olarak geliştirilmiştir.



3. Kullanılacak Araç ve Yöntemler

Proje kapsamında kullanılan temel donanım bileşenleri ve görevleri şunlardır:

- **8086 Mikroişlemci:** Sistemin merkezi işlem birimi olup, oyun algoritmasını yürütür ve tüm birimler arasındaki veri akışını yönetir.
- **74LS373 Latch:** İşlemcinin çoklamalı (multiplexed) adres/veri hattından gelen adres bilgilerini tutarak sistemin kararlı çalışmasını sağlar.
- **8255A Programlanabilir Çevresel Arayüz (PPI):** İşlemci ile dış birimler (LCD, Dot Matrix ve Butonlar) arasındaki veri transferini sağlayan köprü birimidir.
- **LM016L LCD Ekran:** Oyun skoru ve durum mesajlarının kullanıcıya metin tabanlı olarak iletildiği 16x2 karakterlik görüntü birimidir.
- **8x8 Dot Matrix Panel:** Oyunun grafiksel arayüzünü oluşturur; yılanın ve yemlerin konumunu LED'ler aracılığıyla görselleştirir.
- **Yön Butonları (UP, DOWN, LEFT, RIGHT):** Kullanıcının oyun içindeki hareketini belirleyen giriş birimleridir ve Port C üzerinden okunur.

Proje kapsamında **yazılım araçları** da donanım ile entegre şekilde kullanılmaktadır:

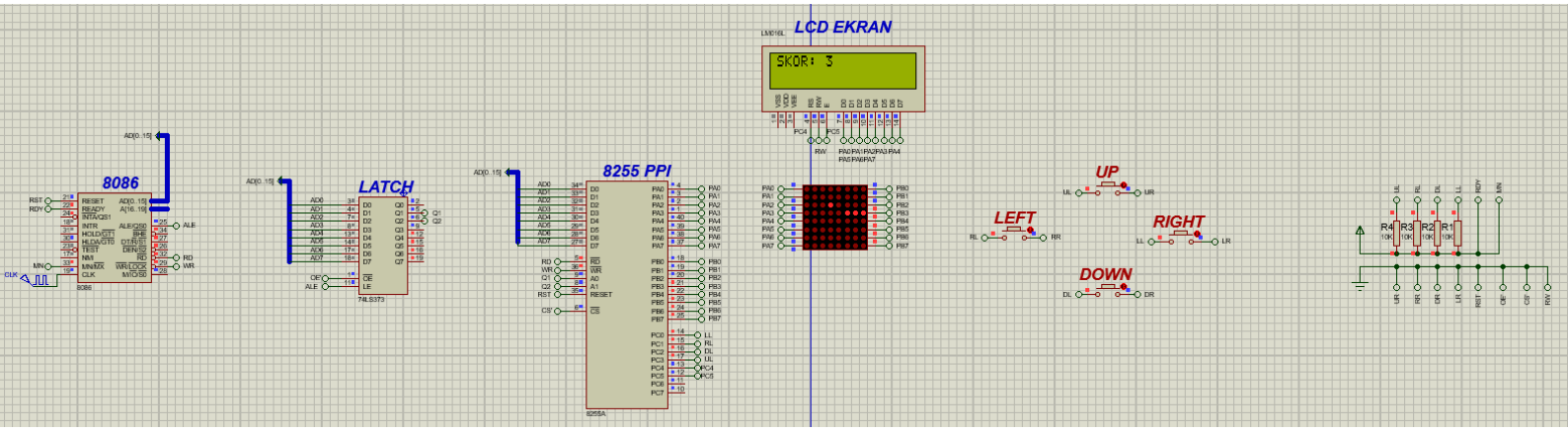
- **EMU8086:** Mikroişlemci programlarının geliştirilmesi ve test edilmesi için kullanılan bir yazılım ortamıdır. 8086 Assembly kodları burada derlenir ve çalıştırılarak algoritmaların doğruluğu kontrol edilir.
- **Proteus Simülasyonu:** Donanım tasarımının sanal ortamda test edilmesini sağlayan simülasyon yazılımıdır. 8086 mikroişlemci, 8255 PPI, LCD ve Dot Matrix gibi tüm bileşenler Proteus üzerinde entegre edilerek devre davranışı gözlemlenebilir.
- **Assembly Dili (8086 Assembly):** Yılan oyunu algoritması, buton okuma, skor hesaplama ve LCD ekran kontrolü gibi temel işlevler bu dil ile programlanmıştır.

Sistem tasarımı; donanım-yazılım entegrasyonu, paralel veri iletimi ve giriş/çıkış portlarının dinamik yönetimi ilkelerini temel almaktadır.

4. SİSTEMİN EVRİMİ VE MİMARİ GELİŞİM

- ## Versiyon 1: Bütünleşik Başlangıç ve Doğrulama Modeli

Bu aşamada sistem, karmaşık bellek birimleri eklenmeden önce mikroişlemci ile temel çevre birimleri arasındaki haberleşmeyi doğrulamak amacıyla tasarlanmış bir **I/O test platformu** niteliğindedir. Bu versiyonun temel hedefi, donanım-yazılım etkileşimini minimal bir mimari üzerinde gözlemlemektir.



☐ Donanım Yapısı ve Yol Ayrıştırması

Sistem; ana kontrol birimi olarak **8086 Mikroişlemci**, adres/veri ayrıştırması için **74LS373 Latch** ve merkezi G/Ç birimi olarak **8255A PPI** entegrelerinden oluşmaktadır.

Adres ve Veri Yolu: 8086'nın AD0–AD15 hatları, **ALE (Address Latch Enable)** sinyali ve 74LS373 entegresi kullanılarak ayrıştırılmıştır. Bu işlem sayesinde adres bilgisi kararlı hale getirilmiş ve 8255A biriminin doğru adreslenmesi sağlanmıştır.

☐ 8255A PPI Yapılandırması ve Birim Kontrolü

Sistemde köprü görevi gören 8255A, **Mode 0** (Basit G/Ç) yapısında programlanmıştır:

- **Port A (PA0–PA7):** Paylaşımlı veri hattı olarak yapılandırılmıştır. Hem LED matrisin satırlarını sürmekte hem de LCD ekranın veri girişlerine (**D0-D7**) karakter/komut bilgisi taşımaktadır.
- **Port B (PB0–PB7):** LED matrisin sütun kontrolünü sağlamak üzere çıkış olarak ayarlanmıştır.
- **Port C (PC0–PC5):** Karma yapıda kullanılmıştır. **PC0-PC3** hatları yön butonlarından (UP, DOWN, LEFT, RIGHT) gelen girişleri okurken; **PC4 (RS)** ve **PC5 (E)** hatları LCD ekranın kontrol sinyallerini üretmektedir.

☐ Görüntüleme Birimleri Entegrasyonu (LCD ve LED Matris)

- **LCD Ekran (LM016L):** 16x2 karakter kapasiteli ekran, sistem durumunu ve skor bilgisini metin tabanlı sunmak için eklenmiştir. Port A üzerinden veri, Port C üzerinden kontrol sinyallerini almaktadır.
- **8x8 LED Matris:** Görsel çıktı sağlamak amacıyla doğrudan Port A ve Port B üzerinden sürülmektedir. Bu aşamada temel amaç, çoklama (multiplexing) tekniği ile port yazma işlemlerinin doğruluğunu test etmektir.

▢ Giriş Birimleri (Buton Yönetimi)

Kullanıcı etkileşimi için tasarlanan dört yön butonu, şemada görüldüğü üzere **pull-down dirençleri** (R1-R4, 10K) ile desteklenerek Port C'ye bağlanmıştır. Butona basılmadığında giriş lojik 0, basıldığında ise lojik 1 seviyesine çekilerek mikroişlemci tarafından polling (sürekli tarama) yöntemiyle algılanmaktadır.

▢ Bellek Yönetimi ve Kısıtlamalar

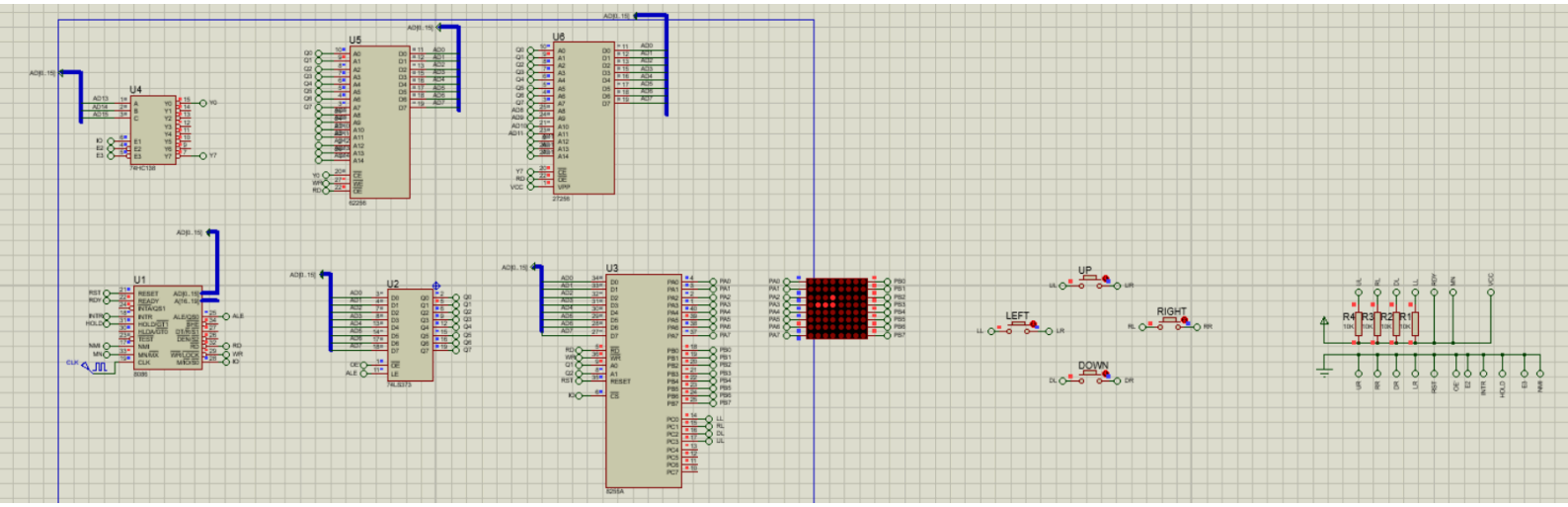
Versiyon 1 aşamasında harici fiziksel RAM ve ROM birimleri kullanılmamıştır. Program kodu Proteus simülasyon ortamının sanal bellek imkanları üzerinden yürütülmüştür. Bu yapı, donanım karmaşıklığını azaltsa da gerçek bir Single Board Computer (SBC) mimarisinde zorunlu olan yonga seçme (chip select) ve tam adres kod çözme mekanizmalarını içermemektedir.

▢ *Sonuç*

Versiyon 1, mikroişlemcinin dış dünya ile iletişim kurabildiğini, LCD ve Matris birimlerinin doğru zamanlamayla sürülebildiğini kanıtlamıştır. Bu yapı, bir oyun sistemi olmaktan ziyade donanım doğrulama platformu görevi görmüş ve Versiyon 2'deki tam mimariye (Bellek + Kod Çözücü) geçiş için sağlam bir temel oluşturmuştur.

• Versiyon 2: RAM ve ROM Kullanılan Modüler Mimari

Projenin bu aşamasında, Versiyon 1'de doğrulanmış olan temel G/Ç yapısı korunarak, sistem literatürde tanımlanan klasik **8086 Minimum Mod** hiyerarşisine sahip gerçek bir mikrobilgisayar mimarisine dönüştürülmüştür. Bu versiyonun temel amacı, mikroişlemcinin harici bellek birimleri ve adres çözümleme mantığı ile tam uyumlu çalıştığı fiziksel altyapıyı doğrulamaktır.



▢ Donanım Genişletmeleri ve Adres Çözümleme

Versiyon 2 ile sisteme dahil edilen ve sistemin "akıllı" birimlerini oluşturan kritik bileşenler şunlardır:

- **74LS138 Adres Dekoderi (U4):** Mikroişlemcinin yüksek anlamlı adres hatlarını (A13-A15) kullanarak, bellek haritasındaki (Memory Map) çakışmaları önler. Sistemdeki her bir birim (RAM, ROM, 8255A) için benzersiz **Chip Select (CS)** sinyalleri üreterek veri yolu trafiğini yönetir.
- **27256 EPROM (U6):** Program kodlarının (firmware) kalıcı olarak saklandığı, 32KB kapasiteli salt okunur bellek birimidir.
- **62256 SRAM (U5):** Oyunun çalışma anındaki değişkenlerini, yılanın koordinat verilerini ve skor bilgisini tutan 32KB kapasiteli dinamik veri belleğidir.
- **Adres Latch Düzenlemesi (U2):** 74LS373 entegresi, 8086'nın multiplexed AD0–AD15 hatlarından adres bilgisini ayrıştırarak tüm bellek birimleri için kararlı bir adres yolu sağlar.

▢ Simülasyon Davranışı ve Teknik Analiz

Şemada görülen donanım tasarımı akademik olarak kusursuzdur; ancak simülasyon ortamında 8086 mimarisinin doğasından kaynaklanan bazı teknik kısıtlamalar gözlemlenmiştir:

1. **Model Soyutlama Farkı:** Proteus içerisindeki 8086 modeli, yüksek performans sağlamak adına bazen harici bus döngülerini (bus cycles) fiziksel zamanlamayla tam eşleyemeyebilir. Bu durum, harici bellek erişimlerinde milisaniyelik senkronizasyon kaymalarına yol açabilmektedir.
2. **Reset Vektörü ve Zamanlama:** 8086, reset sonrası FFFF0H adresinden kod fetch etmeye başlar. Gerçek donanımda bu işlem nanosaniye seviyesinde gerçekleşirken, simülasyonun işlem yükü altında harici ROM'dan veri okuma hızı kararsızlık gösterebilmektedir.
3. **Kararlılık Kararı:** Sistemin yazılımsal doğruluğunu ve oyun algoritmasını hatasız sergileyebilmek adına sunumda Versiyon 1 baz alınmıştır. Versiyon 2 ise sistemin donanımsal omurgasının ve bellek hiyerarşisinin doğruluğunu ispatlamak amacıyla rapora eklenmiştir.

▢ Sonuç

Versiyon 2 şeması, tasarlanan projenin sadece bir yazılımdan ibaret olmadığını, gerçek bir donanım üretimi için gerekli olan adres latching, chip select ve bellek bankası mimarisine tam hakimiyetle geliştirildiğini kanıtlamaktadır. Bu yapı, projenin akademik geçerliliğini ve profesyonel sistem tasarım standartlarını karşılamasını sağlamıştır.

- **Versiyon 1 ile Versiyon 2 Mimari Kıyaslama**

Teknik Parametre	Versiyon 1 (Başlangıç & Test Modeli)	Versiyon 2 (Tam Donanım Mimarisi)
Adres Çözümleme	Yazılımsal/Doğrudan; decoder kullanılmamıştır.	**74LS138** ile donanımsal kod çözme
Program Belleği (ROM)	Proteus dahili hafızası üzerinden yürütülür.	**27256 EPROM**; fiziksel fetch döngüsü sağlar.
Veri Belleği (RAM)	Harici RAM yok; sadece CPU registerları kullanılır.	**62256 SRAM**; dinamik veri saklama alanı.
Simülasyon Kararlılığı	Yüksek; model kısıtlamalarına takılmaz.	Düşük; bus zamanlaması ve model hassasiyeti yüksektir.
Bellek Yapısı	Proteus dahili sanal bellek	Harici Fiziksel SRAM ve EPROM

5. Versiyon 1 Uygulama Senaryosu ve Çalışma Mantığı

Sistemin temel çalışma algoritması; girişlerin taranması, oyun mantığının işlenmesi ve çıktı birimlerinin güncellenmesi olmak üzere üç ana fazdan oluşan sonsuz bir döngü üzerine kurulmuştur.

Donanım Erişimi ve Adresleme Yapısı

Yazılım, 8255A PPI birimini 0F7F0H – 0F7F6H adres aralığında "Memory-Mapped I/O" yöntemiyle kontrol etmektedir. Sistem başlangıcında kontrol register'ına gönderilen 81H komutu ile Port A ve Port B çıkış, Port C'nin alt bitleri ise buton girişlerini okumak üzere giriş olarak tanımlanmıştır.

Oyunun İşleyiş Senaryosu ve Algoritma Akışı

1. Başlatma (Initialization): Program ilk çalıştığında yılanın başlangıç koordinatları, hızı (250 birim) ve boyu (3 birim) veri segmentinde tanımlanır. Ardından LCD ekran temizlenerek "SKOR: 3" bilgisi ekrana yazdırılır.
2. Giriş Tarama (Polling): YON_KONTROL_ET prosedürü, Port C'ye bağlı olan yön butonlarını sürekli olarak tarar. Şemaya uygun olarak PC0 (Sol), PC1 (Sağ), PC2 (Aşağı) ve PC3 (Yukarı) girişlerinden gelen sinyallere göre yılanın hareket vektörü güncellenir.
3. Hız ve Gecikme Yönetimi: Oyun hızı, ZAMAN_SAYACI değişkeni ile kontrol edilir. İşlemci her döngüde matris ekranı tararken aynı zamanda bu sayacı geri saydırarak yılanın hareket hızını simülasyon ortamına uygun şekilde sabitler.
4. Matris Görüntüleme (Multiplexing): 8x8 LED matrisin sürülmesi için Port A üzerinden satır seçimi (tarama) yapılırken, Port B üzerinden GORUNTU_RAM dizisindeki veriler sütunlara aktarılır. NOT AL işlemiyle lojik tersleme yapılarak LED'lerin doğru politede yanması sağlanır.

Oyun Mantığı ve Çarpışma Denetimi

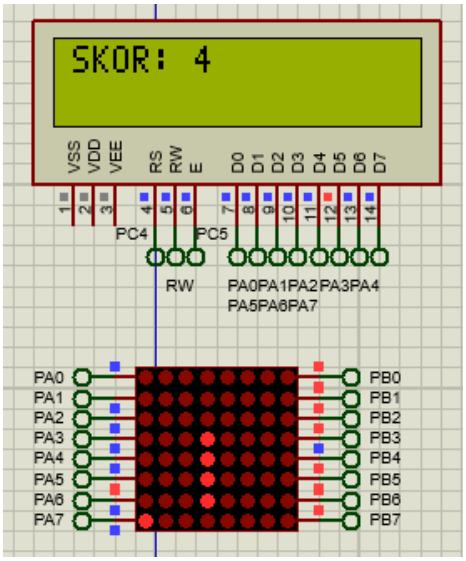
Yılanın her bir hareketiyle birlikte kuyruk segmentleri bir önceki segmentin koordinatını alacak şekilde kaydırılır.

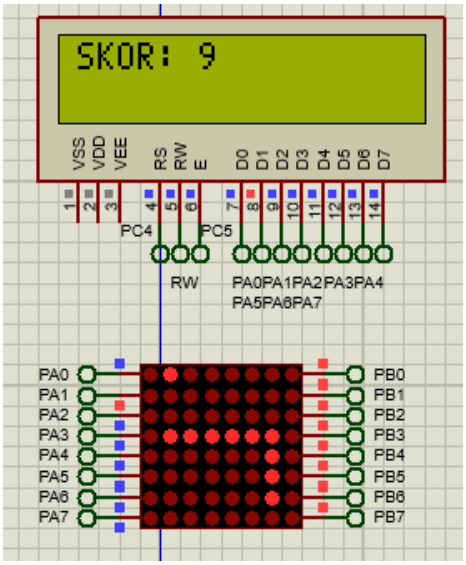
- Yem Mekanizması: Yılanın baş koordinatı ile yem koordinatı çakıştığında yılanın boyu artırılır ve yeni yem koordinatı modüler aritmetik yöntemiyle belirlenerek LCD üzerindeki skor anlık olarak güncellenir.
- Duvar ve Gövde Kontrolü: Yılanın ekran sınırlarını aşması durumunda (0-7 arası sınırlama) yılan karşı taraftan ekrana dahil olur. Ancak yılanın başı kendi gövde koordinatlarından biriyle eşleşirse OYUN_BITTI_GIRIS etiketine dallanarak oyun sonlandırılır.

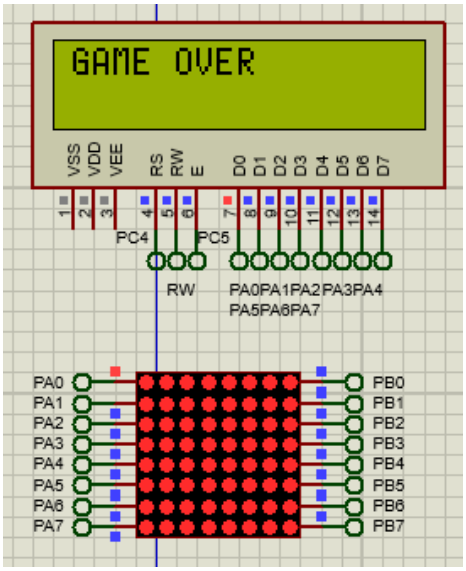
Hata Yönetimi ve Durum Bildirimi

Oyun sona erdiğinde LCD ekran üzerinde "GAME OVER" mesajı görüntülenir. Aynı zamanda LED matris üzerindeki tüm pikseller flaşör (yanıp-sönme) moduna geçirilerek kullanıcıya görsel bir geri bildirim sağlanır. LCD kontrolünde PC4 (RS) ve PC5 (E) pinleri kullanılarak komut ve veri ayırımı donanım şemasına sadık kalınarak gerçekleştirilmektedir.

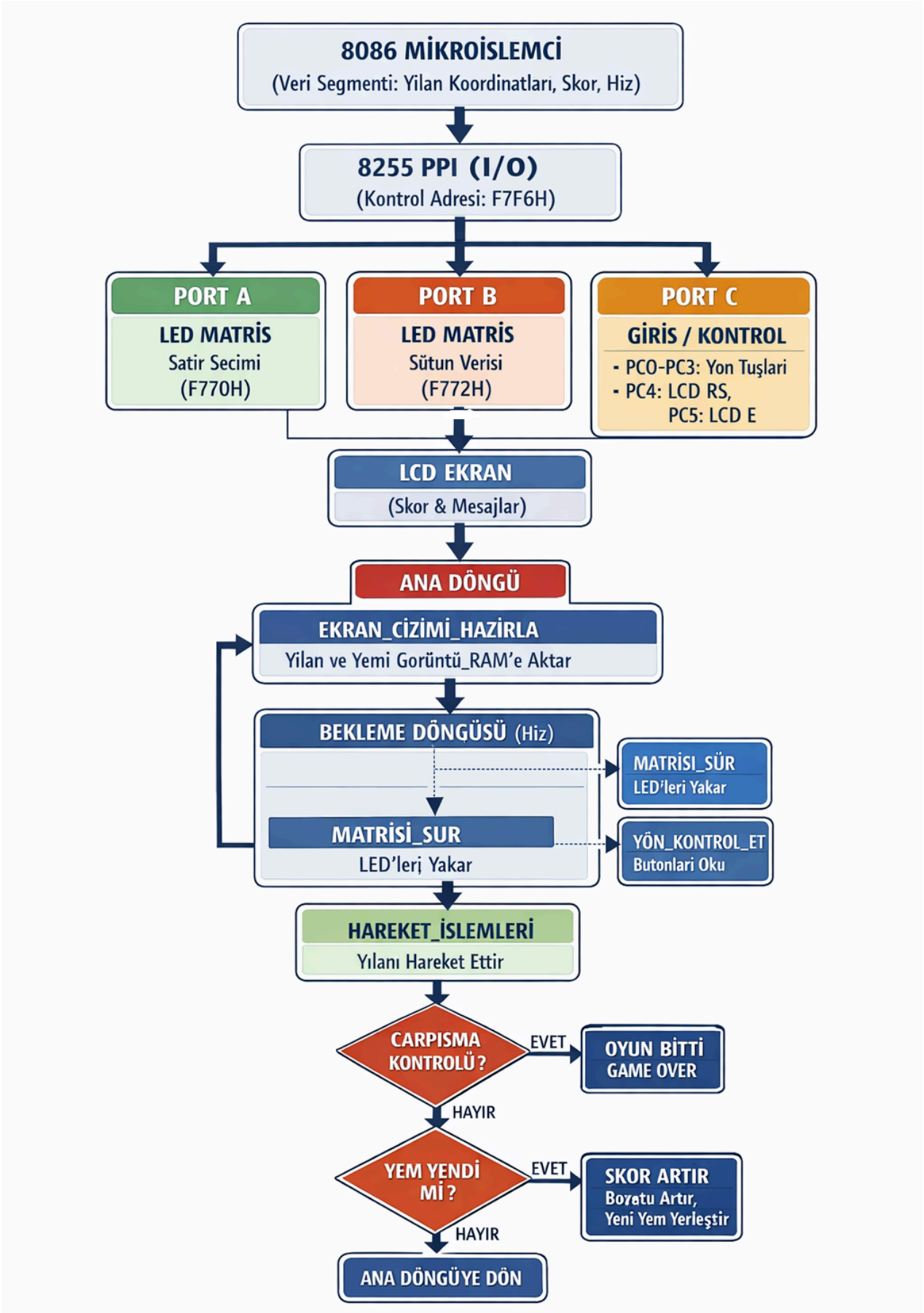
6.Versiyon 1'in Simülasyon Ortamındaki Görselleri







7.Blok Şeması



8. Proje Kodu

```
;=========================================================================

; PROJE: YILAN OYUNU V35 (SON SÜRÜM - PC5 ENABLE FIX)

;=========================================================================

; DONANIM DOĞRULAMASI (image_c6cd23.png):

; RS (Pin 4) -> PC4 (Bit 4)

; RW (Pin 5) -> GND (Toprak)

; E  (Pin 6) -> PC5 (Bit 5)  <--- KESİN BİLGİ

; HIZ        -> 250

; YÖNLER     -> Şemaya göre PC0=Sol, PC1=Sağ, PC2=Aşağı, PC3=Yukarı

;=========================================================================


PORT_A      EQU 0F7F0H

PORT_B      EQU 0F7F2H

PORT_C      EQU 0F7F4H

KONTROL     EQU 0F7F6H


VERI_SEGMENTI SEGMENT

YILAN_X     DB 64 DUP(0)

YILAN_Y     DB 64 DUP(0)

MEVCUT_BOY   DB 3

SUANKI_YON   DB 0


; HIZ AYARI: İsteğin üzerine 250 yapıldı

SABIT_HIZ    DW 250

ZAMAN_SAYACI DW 0
```

```
HEDEF_YEM_X  DB 4

HEDEF_YEM_Y  DB 2

YANIP_SONME_SAYAC DB 0

GORUNTU_RAM  DB 8 DUP(0)

LCD_TEMP     DB 0

VERI_SEGMENTI ENDS


KOD_SEGMENTI SEGMENT

ASSUME CS:KOD_SEGMENTI, DS:VERI_SEGMENTI


BASLANGIC:

MOV AX, VERI_SEGMENTI

MOV DS, AX


; 8255 Kurulumu (A=Out, B=Out, CL=In, CH=Out)

MOV DX, KONTROL

MOV AL, 81H

OUT DX, AL


; LCD Başlat

CALL LCD_INIT

CALL SKOR_GUNCELLE


; Yılan Başlangıç

MOV MEVCUT_BOY, 3

MOV SUANKI_YON, 0       ; Sağa
```

```
MOV YILAN_X[0], 2

MOV YILAN_Y[0], 3

MOV YILAN_X[1], 1

MOV YILAN_Y[1], 3

MOV YILAN_X[2], 0

MOV YILAN_Y[2], 3


ANA_DONGU:

CALL EKRAN_CIZIMI_HAZIRLA

MOV AX, SABIT_HIZ

MOV ZAMAN_SAYACI, AX


BEKLEME_LOOP:

CALL MATRISI_SUR

CALL YON_KONTROL_ET

DEC ZAMAN_SAYACI

JNZ BEKLEME_LOOP

JMP HAREKET_ISLEMLERI


;--------------------------------------------------------------------

; HAREKET

;--------------------------------------------------------------------

HAREKET_ISLEMLERI:

MOV CL, MEVCUT_BOY

DEC CL

MOV CH, 0

MOV SI, CX
```

```
KAYDIRMA:

MOV AL, YILAN_X[SI-1]

MOV YILAN_X[SI], AL

MOV AL, YILAN_Y[SI-1]

MOV YILAN_Y[SI], AL

DEC SI

JNZ KAYDIRMA


CMP SUANKI_YON, 0

JE GIT_SAG

CMP SUANKI_YON, 1

JE GIT_SOL

CMP SUANKI_YON, 2

JE GIT_ASAGI

CMP SUANKI_YON, 3

JE GIT_YUKARI


GIT_SAG:

INC YILAN_X[0]

JMP DUVAR_KONTROL

GIT_SOL:

DEC YILAN_X[0]

JMP DUVAR_KONTROL

GIT_ASAGI:

INC YILAN_Y[0]

JMP DUVAR_KONTROL

GIT_YUKARI:
```

```
DEC YILAN_Y[0]

JMP DUVAR_KONTROL


DUVAR_KONTROL:

MOV AL, YILAN_X[0]

AND AL, 07H

MOV YILAN_X[0], AL

MOV AL, YILAN_Y[0]

AND AL, 07H

MOV YILAN_Y[0], AL


; Çarpışma

MOV CL, MEVCUT_BOY

DEC CL

MOV CH, 0

MOV SI, 1

CARPISMA_TESTI:

MOV AL, YILAN_X[0]

CMP AL, YILAN_X[SI]

JNE TEST_DEVAM

MOV AL, YILAN_Y[0]

CMP AL, YILAN_Y[SI]

JE OYUN_BITTI_GIRIS

TEST_DEVAM:

INC SI

LOOP CARPISMA_TESTI
```

```
; Yem

MOV AL, YILAN_X[0]

CMP AL, HEDEF_YEM_X

JNE ANA_DONGU

MOV AL, YILAN_Y[0]

CMP AL, HEDEF_YEM_Y

JNE ANA_DONGU


INC MEVCUT_BOY

CALL SKOR_GUNCELLE


MOV AL, HEDEF_YEM_X

ADD AL, 3

AND AL, 07H

MOV HEDEF_YEM_X, AL

MOV AL, HEDEF_YEM_Y

ADD AL, 5

AND AL, 07H

MOV HEDEF_YEM_Y, AL

JMP ANA_DONGU


;-------------------------------------------------------------------------

; TUŞ OKUMA (ŞEMAYA GÖRE KESİN)

;-------------------------------------------------------------------------

YON_KONTROL_ET PROC

PUSH AX

PUSH DX
```

```
MOV DX, PORT_C

IN AL, DX

AND AL, 0FH

CMP AL, 0FH

JE YON_CIKIS


; PC0=SOL, PC1=SAĞ, PC2=AŞAĞI, PC3=YUKARI

; Pull-Up (0 gelirse basıldı)


TEST AL, 01H       ; PC0 -> SOL

JZ AYARLA_SOL


TEST AL, 02H       ; PC1 -> SAĞ

JZ AYARLA_SAG


TEST AL, 04H       ; PC2 -> AŞAĞI

JZ AYARLA_ASAGI


TEST AL, 08H       ; PC3 -> YUKARI

JZ AYARLA_YUKARI


JMP YON_CIKIS


AYARLA_SOL:

CMP SUANKI_YON, 0   ; Sağa gidiyorsa dönme

JE YON_CIKIS

MOV SUANKI_YON, 1   ; Sola (Yön 1)
```

```
JMP YON_CIKIS

AYARLA_SAG:
CMP SUANKI_YON, 1   ; Sola gidiyorsa dönme
JE YON_CIKIS
MOV SUANKI_YON, 0   ; Sağa (Yön 0)
JMP YON_CIKIS

AYARLA_ASAGI:
CMP SUANKI_YON, 3   ; Yukarı gidiyorsa dönme
JE YON_CIKIS
MOV SUANKI_YON, 2   ; Aşağı (Yön 2)
JMP YON_CIKIS

AYARLA_YUKARI:
CMP SUANKI_YON, 2   ; Aşağı gidiyorsa dönme
JE YON_CIKIS
MOV SUANKI_YON, 3   ; Yukarı (Yön 3)
JMP YON_CIKIS

YON_CIKIS:
POP DX
POP AX
RET
YON_KONTROL_ET ENDP

;-------------------------------------------------------------------
```

```
; OYUN BİTİŞ

;-------------------------------------------------------------------------

OYUN_BITTI_GIRIS:

CALL GAME_OVER_YAZ



FLASOR_LOOP:

MOV CX, 8

MOV SI, 0

DOL: MOV GORUNTU_RAM[SI], 0FFH

INC SI

LOOP DOL

MOV CX, 20

DOL_WAIT:

PUSH CX

CALL MATRISI_SUR

POP CX

LOOP DOL_WAIT

MOV CX, 8

MOV SI, 0

BOS: MOV GORUNTU_RAM[SI], 00H

INC SI

LOOP BOS

MOV CX, 20

BOS_WAIT:

PUSH CX

CALL MATRISI_SUR

POP CX
```

```
LOOP BOS_WAIT

JMP FLASOR_LOOP



;---------------------------------------------------------------------

; EKRAN ÇİZİM

;---------------------------------------------------------------------

EKRAN_CIZIMI_HAZIRLA PROC

PUSH AX

PUSH BX

PUSH CX

PUSH SI

PUSH DI

MOV CX, 8

MOV SI, 0

TEMIZLE_LP: MOV GORUNTU_RAM[SI], 0

INC SI

LOOP TEMIZLE_LP

INC YANIP_SONME_SAYAC

TEST YANIP_SONME_SAYAC, 1

JZ YILAN_CIZ_L

MOV BL, HEDEF_YEM_X

MOV BH, HEDEF_YEM_Y

MOV SI, 0

MOV AL, BH

CBW

MOV SI, AX

MOV CL, BL
```

```
MOV AL, 1

SHL AL, CL

OR GORUNTU_RAM[SI], AL

YILAN_CIZ_L:

MOV CL, MEVCUT_BOY

MOV CH, 0

MOV DI, 0

YILAN_LOOP_L:

MOV AL, YILAN_Y[DI]

CBW

MOV SI, AX

MOV DL, YILAN_X[DI]

MOV AL, 1

PUSH CX

MOV CL, DL

SHL AL, CL

POP CX

OR GORUNTU_RAM[SI], AL

INC DI

LOOP YILAN_LOOP_L

POP DI

POP SI

POP CX

POP BX

POP AX

RET

EKRAN_CIZIMI_HAZIRLA ENDP
```

```
;-------------------------------------------------------------------------

; MATRİS SÜRME

;-------------------------------------------------------------------------

MATRISI_SUR PROC

PUSH AX

PUSH BX

PUSH CX

PUSH SI

PUSH DX


; LCD Kontrol pinlerini SIFIRLA

MOV DX, PORT_C

MOV AL, 0

OUT DX, AL


MOV CX, 8

MOV SI, 0

MOV BH, 01H

TARAMA_L:

MOV DX, PORT_A

MOV AL, BH

OUT DX, AL


MOV DX, PORT_B

MOV AL, GORUNTU_RAM[SI]

NOT AL
```

```
OUT DX, AL

PUSH CX
MOV CX, 50
BEKLE_L: NOP
LOOP BEKLE_L
POP CX

MOV DX, PORT_A
MOV AL, 0
OUT DX, AL

ROL BH, 1
INC SI
LOOP TARAMA_L

POP DX
POP SI
POP CX
POP BX
POP AX
RET
MATRISI_SUR ENDP

;=========================================================================
; LCD FONKSİYONLARI (PC4=RS, PC5=E - SON KONTROL)
;=========================================================================
```

```
GECIKME_LCD PROC NEAR
PUSH CX
MOV CX, 500
DLY_LCD: NOP
LOOP DLY_LCD
POP CX
RET
GECIKME_LCD ENDP

; Komut (RS=0)
LCD_KOMUT PROC NEAR
PUSH DX
PUSH AX
MOV LCD_TEMP, AL

; 1. Veriyi Port A'ya yaz
MOV DX, PORT_A
MOV AL, LCD_TEMP
OUT DX, AL

; 2. Sinyal: E=1 (PC5), RS=0 (PC4) -> 0010 0000b = 20H
MOV DX, PORT_C
MOV AL, 20H
OUT DX, AL
NOP
NOP
```

```
; 3. Sinyal: E=0 (PC5), RS=0 -> 00H
MOV AL, 00H
OUT DX, AL

CALL GECIKME_LCD
POP AX
POP DX
RET
LCD_KOMUT ENDP

; Veri (RS=1)
LCD_VERI PROC NEAR
PUSH DX
PUSH AX
MOV LCD_TEMP, AL

; 1. Veriyi Port A'ya yaz
MOV DX, PORT_A
MOV AL, LCD_TEMP
OUT DX, AL

; 2. Sinyal: E=1 (PC5), RS=1 (PC4) -> 0011 0000b = 30H
MOV DX, PORT_C
MOV AL, 30H
OUT DX, AL
NOP
NOP
```

```
; 3. Sinyal: E=0 (PC5), RS=1 (PC4) -> 0001 0000b = 10H
MOV AL, 10H
OUT DX, AL

CALL GECIKME_LCD
POP AX
POP DX
RET
LCD_VERI ENDP

LCD_INIT PROC NEAR
PUSH AX
PUSH DX
MOV AL, 38H
CALL LCD_KOMUT
MOV AL, 0CH
CALL LCD_KOMUT
MOV AL, 01H
CALL LCD_KOMUT
MOV AL, 06H
CALL LCD_KOMUT
POP DX
POP AX
RET
LCD_INIT ENDP
```

```
SKOR_GUNCELLE PROC NEAR
PUSH AX
MOV AL, 80H
CALL LCD_KOMUT

MOV AL, 'S'
CALL LCD_VERI
MOV AL, 'K'
CALL LCD_VERI
MOV AL, 'O'
CALL LCD_VERI
MOV AL, 'R'
CALL LCD_VERI
MOV AL, ':'
CALL LCD_VERI
MOV AL, ' '
CALL LCD_VERI

MOV AL, MEVCUT_BOY
ADD AL, 30H
CALL LCD_VERI
POP AX
RET
SKOR_GUNCELLE ENDP

GAME_OVER_YAZ PROC NEAR
MOV AL, 01H
```

```
CALL LCD_KOMUT

MOV AL, 'G'
CALL LCD_VERI
MOV AL, 'A'
CALL LCD_VERI
MOV AL, 'M'
CALL LCD_VERI
MOV AL, 'E'
CALL LCD_VERI
MOV AL, ' '
CALL LCD_VERI
MOV AL, 'O'
CALL LCD_VERI
MOV AL, 'V'
CALL LCD_VERI
MOV AL, 'E'
CALL LCD_VERI
MOV AL, 'R'
CALL LCD_VERI
RET
GAME_OVER_YAZ ENDP

KOD_SEGMENTI ENDS
END BASLANGIC
```