به نام خدا

نام درس : ریزپردازنده و زبان اسمبلی

نام پروژه: بازی دوز

اعضای گروه : زینب باقری، مهلا عزیزی، الهه طاهری

استاد مربوطه : دکتر قزوینی

هدف پروژه:

هدف این پروژه، طراحی و پیاده‌سازی بازی دوز با استفاده از پردازنده 8086 و دو آی‌سی 8255 برای ارتباط با ورودی و خروجی است. در این پروژه، بازیکنان با فشردن کلیدها حرکات خود را ثبت می‌کنند و وضعیت بازی از طریق LEDها نمایش داده می‌شود. منطق بازی شامل کنترل نوبت بازیکنان، جلوگیری از حرکات نامعتبر، تشخیص برنده یا مساوی و نمایش نتایج به‌صورت بصری است. این پروژه تمرینی برای آشنایی عملی با برنامه‌نویسی اسمبلی، ارتباط با سخت‌افزار و مفاهیم پایه سیستم‌های دیجیتال است.

توضیح قسمت سخت افزار:

در ابتدا به معرفی آی‌سی‌ها و تراشه‌های اصلی این پروژه می‌پردازیم:

1-میکروپروسسور 8086: میکروپروسسور 8086، محصول شرکت اینتل، یک پردازنده 16 بیتی است که در سال 1978 معرفی شد. این تراشه به‌عنوان قلب پردازشی پروژه عمل می‌کند و مسئولیت اجرای برنامه بازی، پردازش ورودی‌ها و کنترل خروجی‌ها را بر عهده دارد.

مشخصات فنی:

معماری: 16 بیتی با گذرگاه داده 16 بیتی و گذرگاه آدرس 20 بیتی (پشتیبانی از 1 مگابایت حافظه)

پایه‌های کلیدی:

- AD0-AD15 (پایه‌های 2-16، 39): گذرگاه آدرس/داده برای انتقال داده و آدرس
- A16-A19 (پایه‌های 35-38): خطوط آدرس برای انتخاب حافظه یا دستگاه‌های جانبی
- RESET (پایه 21): برای ریست پردازنده و شروع اجرای برنامه از آدرس h0000
- IO/M (پایه 28): مشخص‌کننده نوع عملیات (ورودی/خروجی یا حافظه)
- RD (پایه 32): سیگنال خواندن داده
- WR (پایه 29): سیگنال نوشتن داده

کاربرد در پروژه:

- اجرای برنامه اسمبلی بازی دوز
- مدیریت ورودی‌ها از کلیدها (از طریق 8255) و ارسال خروجی به ال‌ای‌دی‌ها
- تولید آدرس‌ها برای انتخاب تراشه‌های 8255 با کمک دیکدر 74LS138

2-تراشه 74LS373 (لچ 8 بیتی): تراشه 74LS373 یک لچ 8 بیتی شفاف (Octal D-Type Transparent Latch) است که برای ذخیره موقت داده‌ها یا آدرس‌ها در سیستم‌های دیجیتال استفاده می‌شود.

پایه‌های کلیدی:

- D0-D7 (پایه‌های 3، 4، 7، 8، 13، 14، 17، 18): ورودی‌های داده
- Q0-Q7 (پایه‌های 2، 5، 6, 9، 12, 15, 16, 19): خروجی‌های داده
- LE (پایه 11): سیگنال فعال‌سازی لچ (فعال-بالا)
- OE (پایه 1): سیگنال فعال‌سازی خروجی (فعال-پایین)

کاربرد در پروژه:

- ذخیره آدرس‌های تولیدشده توسط 8086 برای جلوگیری از تداخل در گذرگاه آدرس/داده
- کمک به جداسازی آدرس‌ها برای انتخاب تراشه‌های 8255 توسط دیکدر 74LS138

3-تراشه 8255: تراشه 8255 (Programmable Peripheral Interface) یک رابط ورودی/خروجی قابل‌برنامه‌ریزی است که برای اتصال دستگاه‌های خارجی (مثل کلیدها و ال‌ای‌دی‌ها) به میکروپروسسور استفاده می‌شود و از رجیسترها تشکیل شده است. در این پروژه، دو تراشه 8255 (1# و 2#) برای مدیریت ورودی‌ها و خروجی‌ها به کار رفته‌اند.

مشخصات فنی:

پورت‌ها: سه پورت 8 بیتی (A، B، C) که به‌صورت ورودی یا خروجی قابل‌تنظیم‌اند.

پایه‌های کلیدی:

- D0-D7 (پایه‌های 27-34): گذرگاه داده برای انتقال داده به/از 8086
- CS (پایه 6): سیگنال انتخاب تراشه (فعال-پایین)
- A0، A1 (پایه‌های 9، 10): برای انتخاب پورت A، B، C، یا ثبت کنترل
- PA0-PA7 (پایه‌های 30-37): پورت A
- PB0-PB7 (پایه‌های 18-25): پورت B
- PC0-PC7 (پایه‌های 14-17، 1-4): پورت C
- RESET (پایه 35): برای تنظیم پورت‌ها به حالت ورودی در شروع

کاربرد در پروژه:

8255 اول:

پورت A (خروجی): برای کنترل 8 ال‌ای‌دی دو رنگ

پورت B (خروجی): برای کنترل 8 ال‌ای‌دی دو رنگ

پورت C (خروجی): برای کنترل ال‌ای‌دی دو رنگ نهم

8255 دوم:

پورت A (ورودی): برای خواندن 8 کلید انتخاب خانه‌ها

پورت B (ورودی): برای خواندن 8 کلید انتخاب خانه‌ها

نقش در پروژه:

- دریافت ورودی از کلیدها برای انتخاب خانه‌ها و تأیید حرکت
- کنترل ال‌ای‌دی‌های دو رنگ برای نمایش وضعیت صفحه بازی (X، O، یا خالی)

4-تراشه 74LS138 (دیکدر 3 به 8): تراشه 74LS138 یک دیکدر 3 به 8 است که برای انتخاب دستگاه‌های جانبی با استفاده از خطوط آدرس استفاده می‌شود.

پایه‌های کلیدی:

- A0-A2 (پایه‌های 1-3): ورودی‌های انتخاب آدرس
- G2A، G2B (پایه‌های 4، 6): سیگنال‌های کنترلی فعال-پایین
- G1 (پایه 5): سیگنال کنترلی فعال-بالا
- Y0-Y7 (پایه‌های 15-7): خروجی‌های فعال-پایین

کاربرد در پروژه:

- انتخاب تراشه‌های 8255 (1# و 2#) با استفاده از خطوط آدرس (A0-A2 یا A2-A4)
- تولید سیگنال CS (Chip Select) برای فعال‌سازی 8255ها

خروجی Y0 به 1# 8255 CS و خروجی Y4 به 2# 8255 CS وصل است.

با دریافت آدرس از 8086 (از طریق 74LS373)، مشخص می‌کند کدام 8255 فعال شود تا پورت‌های آن (A، B، C، یا ثبت کنترل) قابل دسترسی باشند.

اتصالات:

در ادامه، جزئیات اتصالات شرح داده می‌شود.

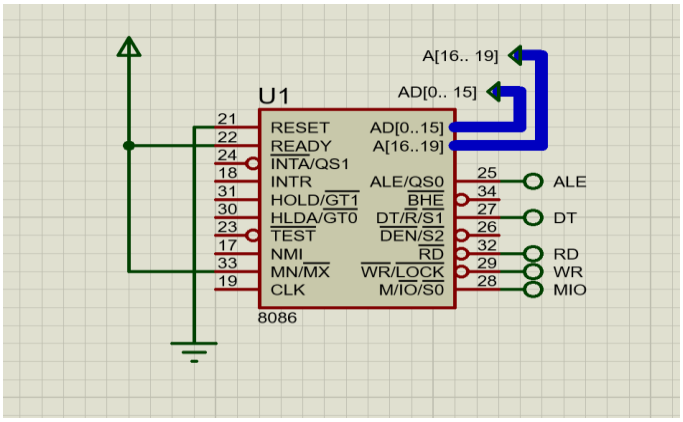
1. اتصالات میکروپروسسور 8086

IO/M (پایه 28): به پایه G2A دیکدر 74LS138 متصل است تا عملیات ورودی/خروجی را از دسترسی حافظه جدا کند.

RD (پایه 32): به پایه RD تراشه‌های 8255

WR (پایه 29): به پایه WR تراشه‌های 8255

RESET (پایه 21): به GND



2. اتصالات تراشه 74LS373 (لچ 8 بیتی)

ورودی‌های داده (D0-D7):

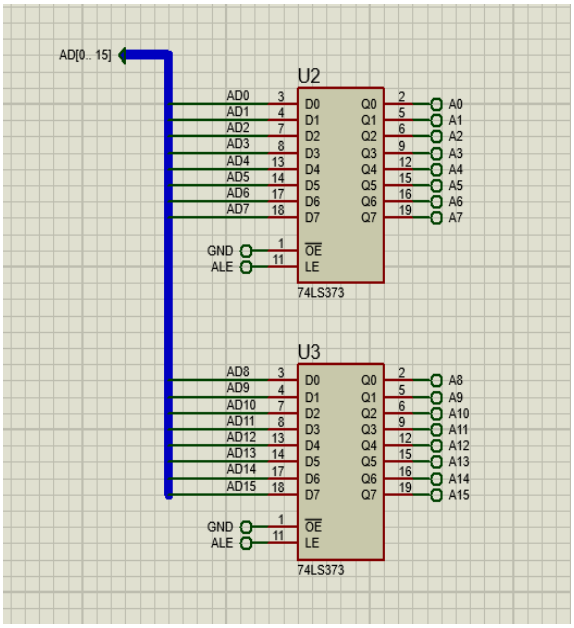
پایه‌های 3، 4، 7، 8، 13، 14، 17، 18 در دو تراشه به AD0-AD7 گذرگاه آدرس/داده 8086 متصل هستند.

خروجی‌های داده (Q0-Q7):

پایه‌های 2، 5، 6، 9، 12، 15، 16، 19 در دو تراشه به ورودی‌های آدرس دیکدر 74LS138 (A0-A2) و A3 به G1 دیکدر و پایه‌های A4، A5 هر دو به 8255 متصل هستند.

LE (پایه 11): به سیگنال ALE (Address Latch Enable، پایه 25) 8086 متصل است تا آدرس‌ها را لچ کند.

OE (پایه 1): به زمین (فعال-پایین) برای فعال بودن دائمی خروجی‌ها



3. اتصالات تراشه 74LS138 (دیکدر 3 به 8)

ورودی‌های آدرس (A0-A2):

پایه یک (A0): به A0 از خروجی 74LS373

پایه دو (A1): به A1 از خروجی 74LS373

پایه سه (A2): به A2 از خروجی 74LS373

G2A (پایه چهار): به IO/M (فعال-پایین) برای فعال شدن در عملیات ورودی/خروجی

G2B (پایه شش): به زمین (فعال-پایین)

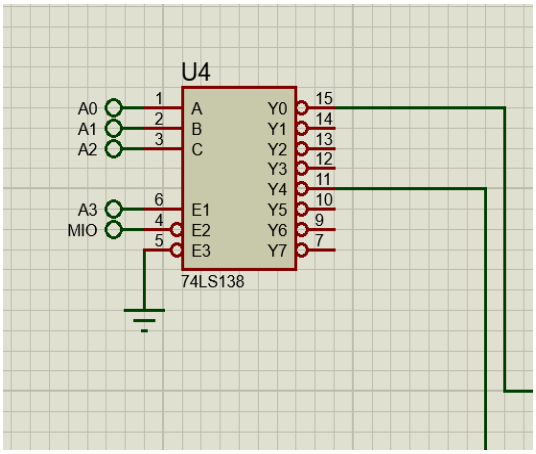
G1 (پایه پنج): به A3 از خروجی 74LS373

Y0 (پایه 15): به پایه CS (پایه 6) 8255 اول

Y4 (پایه 11): به پایه CS (پایه 6) 8255 دوم

نقش:

دیکدر 74LS138 با دریافت خطوط آدرس (A2-A4) از 74LS373، سیگنال CS را برای انتخاب یکی از 8255ها (Y0 برای 8255 اول و Y4 برای 8255 دوم) تولید می‌کند.



4. اتصالات تراشه‌های 8255

دو تراشه 8255 (اول و دوم) برای مدیریت ورودی‌ها و خروجی‌ها استفاده شده‌اند. اتصالات هر تراشه به‌صورت زیر است:

1.4. تراشه 8255 اول

گذرگاه داده (D0-D7):

پایه‌های 27-34 به گذرگاه داده AD0-AD7 8086 متصل هستند.

انتخاب پورت (A0، A1):

A0 (پایه نه): به A4 از خروجی 74LS373

A1 (پایه 10): به A5 از خروجی 74LS373

پورت‌ها:

هر سه پورت این تراشه به ال‌ای‌دی‌های دورنگ متصل هستند.

RESET (پایه 35): به زمین

2.4. تراشه 8255 دوم

گذرگاه داده (D0-D7):

پایه‌های 27-34 به گذرگاه داده AD0-AD7 8086 متصل هستند.

انتخاب پورت (A0، A1):

همانند تراشه 8255 اول

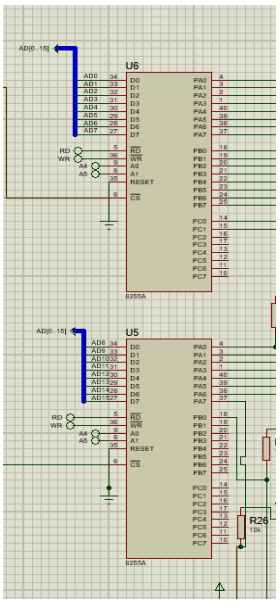
انتخاب تراشه (CS):

پورت‌ها:

دو پورت به کلیدها متصل هستند. هر کلید با مقاومت پول‌آپ kΩ10 به +V5 و طرف دیگر به زمین. فشار دادن کلید، پایه مربوطه (مثل PB0) را به صفر منطقی می‌رساند.

RESET (پایه 35): به زمین

در 8255، A0 و A1 برای انتخاب پورت‌های A، B، C، یا ثبت کنترل استفاده می‌شوند.



ملاحظات طراحی:

مقاومت‌های پول‌آپ: مقاومت‌های kΩ10 برای کلیدها از شناور شدن پایه‌های ورودی (PB0-PB7، PC0-PC1) جلوگیری می‌کنند.

مقاومت‌های سری ال‌ای‌دی‌ها: مقاومت‌های Ω330 جریان ال‌ای‌دی‌ها را محدود می‌کنند تا از آسیب دیدن آن‌ها جلوگیری شود.

جدا کردن آدرس‌ها: لچ 74LS373 از تداخل آدرس و داده در گذرگاه مشترک 8086 جلوگیری می‌کند.

توضیح قسمت کد:

بخش 1: تعریف آدرس پورت‌ها

در این بخش، آدرس پورت‌های آی‌سی‌های 8255 مشخص شده‌اند. هر 8255 دارای سه پورتA ، B، C و یک رجیستر کنترل است. با استفاده از دیکدر F13874، آدرس‌دهی از طریق خروجی‌های Y0 تا Y7 انجام شده:

asm

```
PORTA_1    EQU 00h
PORTB_1    EQU 02h
PORTC_1    EQU 04h
CTRL_1     EQU 06h
PORTA_2    EQU 08h
PORTB_2    EQU 0Ah
PORTC_2    EQU 0Ch
CTRL_2     EQU 0Eh
```

بخش 2: مقداردهی اولیه سگمنت‌ها

در ابتدای برنامه، مقادیر اولیه برای رجیسترهای سگمنت داده و اضافه تنظیم می‌شود تا به متغیرها و حافظه‌ی داده‌ها دسترسی داشته باشیم:

asm

```
MOV AX, 0
MOV DS, AX
MOV ES, AX
```

بخش 3: تنظیم مود کاری 8255

در این مرحله، مود کاری دو آی‌سی 8255 تنظیم می‌شود. در هر دو مورد، پورت A به عنوان ورودی و پورت‌های B و C به عنوان خروجی تنظیم شده‌اند:

asm

MOV AL, 10000000b

OUT CTRL_1, AL

MOV AL, 10011011b

OUT CTRL_2, AL

بخش 4: تعریف متغیرهای اصلی بازی

دو متغیر تعریف شده‌اند:

BOARD *برای ذخیره وضعیت نه خانه‌ی بازی دوز = 0) خالی، X1 = ، 2(O =

PLAYER *برای نگهداری نوبت بازیکن فعلی (1 یا 2)

asm

BOARD DB 9 DUP(0)

PLAYER DB 1

بخش 5: حلقه‌ی اصلی بازی

در حلقه‌ی اصلی، توابع مختلف بازی صدا زده می‌شوند تا روند بازی به‌طور پیوسته اجرا شود:

asm

MAIN:

CALL READ_INPUT

CALL UPDATE_BOARD

CALL DISPLAY_BOARD

CALL CHECK_WINNER

CALL CHECK_TIE

JMP MAIN

بخش 6: دریافت ورودی از بازیکن

با خواندن مقدار از پورتA ، بررسی می‌شود که کدام کلید فشرده شده. اگر هیچ‌کدام از هشت کلید اول فعال نباشند، کلید نهم از پورت B چک می‌شود. مقدار کلید فشرده در رجیستر SI قرار می‌گیرد:

asm

READ_INPUT:

IN AL, PORTA_2

MOV BL, AL

MOV CX, 8

MOV SI, 0

CHECK_KEY:

TEST BL, 1

JNZ KEY_PRESSED

SHR BL, 1

INC SI

LOOP CHECK_KEY

IN AL, PORTB_2

TEST AL, 1

JNZ KEY_PRESSED_9

RET

بخش 7: ثبت حرکت بازیکن

اگر خانه انتخاب‌شده خالی باشد، مقدار آن برابر با شماره بازیکن قرار می‌گیرد (1 یا 2). اگر خانه قبلاً اشغال شده باشد، حرکت نامعتبر است:

asm

KEY_PRESSED:

CMP BOARD[SI], 0

JNE INVALID_MOVE

MOV AL, PLAYER

MOV BOARD[SI], AL

JMP SWITCH_PLAYER

KEY_PRESSED_9:

```
MOV SI, 8
CMP BOARD[SI], 0
JNE INVALID_MOVE
MOV AL, PLAYER
MOV BOARD[SI], AL
```

بخش 8: تعویض نوبت بازیکن

پس از ثبت حرکت، نوبت بازیکن تغییر داده می‌شود. اگر نوبت بازیکن 1 باشد، به 2 تغییر داده می‌شود و بالعکس:

```
asm
SWITCH_PLAYER:
CMP PLAYER, 1
JE SET_PLAYER_O
MOV PLAYER, 1
JMP END_INPUT
SET_PLAYER_O:
MOV PLAYER, 2
END_INPUT:
RET
```

بخش 9: هشدار برای خطا

زمانی که کلیدی فشار داده شده است ولی خانه مربوطه قبالپر شده است. درنتیجه ال ای دی ها روشن میشوند تا هشدار دهند، این درواقع یک هشدار بصری برای بازیکن است که حرکتش مجاز نبوده است.

```
INVALID_MOVE:
MOV AL, 0FFh
OUT PORTA_1, AL
OUT PORTA_2, AL
RET
```

بخش 10: ثبت حرکت درخانه

```
UPDATE_BOARD:
CMP SI, 9
JGE END_UPDATE
CMP BOARD[SI], 0
JE END_UPDATE
MOV AL, BOARD[SI]
CMP AL, PLAYER
```

بخش 11: نمایش وضعیت در خانه

هر خانه از نه خانه با دو LED نشان داده می شود:

اگر مقدارش یک باشد (بازی کن X): یک LED قرمز روشن می شود.

اگر مقدارش دو باشد (بازی کن O): یک LED سبز روشن می شود.

اگر مقدارش صفر باشد: LED خاموش میماند.

```
:DISPLAY_LOOP
CMP SI, 8
JGE DISPLAY_LAST
CMP BOARD[SI], 1 ; X
JE SET_RED
CMP BOARD[SI], 2 ; O
JE SET_GREEN
JMP NEXT_LED
```

بخش 12: نمایش رنگ ال ای دی ها

برای هر خانه در آرایه بازی(BOARD)، اگر مقدارش:

1باشد → بایدLED قرمز روشن شود.

2باشد → بایدLED سبز روشن شود.

هرخانه با دو بیت نمایش داده میشود (یکی برای قرمز،یکی برای سبز).

```
:SET_RED
MOV AX, SI
```

```
SHL AL, 1
MOV CL, AL
MOV BL, 1
SHL BL, CL
OR AL, BL
JMP NEXT_LED
SET_GREEN:
MOV AX, SI
SHL AL, 1
INC AL
MOV CL, AL
MOV BL, 1
SHL BL, CL
OR AL, BL
NEXT_LED:
INC SI
CMP SI, 9
JL DISPLAY_LOOP
OUT PORTA_1, AL
RET
```

بخش 13:نمایش خانه نهم

(خانه شماره هشت در آرایه BOARD) روی پورت جداگانه (PORTA_2)

اگر خانه نهم متعلق بهX باشد:

```
:END_LED_9
OUT PORTA_2, AL
```

اگر خانه نهم متعلق بهO باشد:

```
:SET_GREEN_9
OR AL, 00000010b
:DISPLAY_LAST
MOV AL, 0
CMP BOARD[8], 1
```

```
JE SET_RED_9
CMP BOARD[8], 2
JE SET_GREEN_9
JMP END_LED_9
:SET_RED_9
OR AL, 00000001b
JMP END_LED_9
:SET_GREEN_9
OR AL, 00000010b
:END_LED_9
OUT PORTA_2, AL
RET
```

بخش 14: تشخیص برنده

بررسی میکند که آیا بازیکنی برنده شده است یانه. فقط ردیف های افقی(سطرها) را بررسی میکند و اگر سه خانه پشت سرهم مقداریکسانی داشته باشند (و خالی نباشند)، آن بازیکن را برنده اعلام میکند.

```
:CHECK_WINNER
MOV SI, 0
:CHECK_ROWS
MOV AL, BOARD[SI]
CMP AL, 0
JE NEXT_ROW
CMP AL, BOARD[SI+1]
JNE NEXT_ROW
CMP AL, BOARD[SI+2]
JE WIN_FOUND
:NEXT_ROW
ADD SI, 3
CMP SI, 9
JL CHECK_ROWS
```

```
RET
WIN_FOUND
MOV AL, 0FFh
OUT PORTA_1, AL
OUT PORTA_2, AL
RET
```

بخش 15: بررسی تساوی

تمام خانه های بازی را بررسی میکند اگرهمه خانه ها پربودند (هیچ`0` وجود نداشت)،با روشن کردن ال ای دی ها تساوی اعلام میشود.اگرحداقل یک خانه خالی بود، کاری انجام نمیدهد.

```
:CHECK_TIE
MOV SI, 0
:CHECK_FULL
CMP BOARD[SI], 0
JE NOT_FULL
INC SI
CMP SI, 9
JL CHECK_FULL
MOV AL, 0FFh
OUT PORTA_1, AL
OUT PORTA_2, AL
RET
:NOT_FULL
RET
:ENDLESS
JMP ENDLESS
CODE ENDS
END START
```

نتیجه‌گیری:

در این پروژه با استفاده از زبان اسمبلی و پردازنده 8086، موفق شدیم یک نمونه ساده اما کاربردی از بازی دوز را طراحی و پیاده‌سازی کنیم. با بهره‌گیری از آی‌سی‌های 8255 و LED ها، توانستیم ارتباط بین ورودی (کلیدها) و خروجی (نمایش وضعیت بازی) را به صورت سخت‌افزاری پیاده کنیم. این پروژه توانایی ما را در کار با پورت‌های ورودی/خروجی، برنامه‌نویسی سطح پایین، و تحلیل منطق بازی تقویت کرد. همچنین نشان داد که می‌توان حتی با سخت‌افزارهای ساده و زبان‌های سطح پایین، پروژه‌های تعاملی و جذاب طراحی نمود.