فصل ششم
روش های پایه ی ورودی/خروجی

مقدمه

پورت‌های I/O مسیری برای عبور داده‌های مبادله شده بین ریزپردازنده و دستگاههای جانبی است.

روش‌های I/O

- **موازی**: ساده‌ترین راه استفاده از پورت موازی است. در این روش تمامی بیت‌هایی که یک کلمه را تشکیل می‌دهند، با هم وارد یا خارج می‌شوند.
- **سریال**: بیت‌های داده همگی در یک خط قرار می‌گیرند و یکی یکی از آن خط منتقل می‌شوند. طبیعتا روش سریال کندتر از موازی است.

مزایای استفاده از روش سریال:

ساختن یک کانال دو طرفه‌ی همگام (فرستادن و دریافت داده همگام انجام پذیر است) با استفاده از ۳ سیم هادی امکان‌پذیر است که یکی برای فرستادن، دیگری برای دریافت و سومی هم برای مشترک کردن زمین بین فرستنده و گیرنده است.

بیت‌های سریال را می‌توان با استفاده از مودم به سیگنال آنالوگ تبدیل کرد و از طریق خطوط تلفن منتقل کرد. مودم گیرنده سیگنال صوتی را مجددا به **0** و **1** تبدیل می‌کند. بدین ترتیب کامپیوتر می‌تواند با سیستمی در فاصله‌ی چند هزار کیلومتری ارتباط برقرار کند.

I/O موازی

- سخت‌افزار لازم برای پورت **I/O** موازی شبیه مدار واسط **ROM** و **RAM** است.
- زمانیکه **CPU** یک دستور خروجی را اجرا می‌کند (سیکل نوشتن **I/O**)، پورت باید داده موجود بر باس را ذخیره کند.
- به طور مشابه وقتی دستور ورودی اجرا می‌شود (سیکل خواندن **I/O**) پورت باید داده را بر خطوط باس داده وارد کند.
- پورت‌های **I/O** آدرس‌های ویژه خود را دارند.

I/O موازی

۸۰۸۸ و ۸۰۸۶ فقط دو دستور برای ورود و خروج داده دارد.

- ✓ "شماره‌ی پورت ,IN AX " : برای خواندن از پورت
- ✓ " AL,شماره‌ی پورتOUT " برای نوشتن در پورت

برای هر کدام دو قالب مستقیم و غیرمستقیم وجود دارد.

- در **قالب مستقیم** شماره‌ی پورت در دستور ذکر می‌شود و لذا می‌تواند ۲۵۶ پورت را آدرس‌دهی کند.

- در **قالب غیرمستقیم** رجیستر DX آدرس پورت را نگه‌می‌دارد و بنابراین می‌تواند ۶۵۵۳۶ پورت را آدرس‌دهی کند.

I/Oموازی (ادامه)

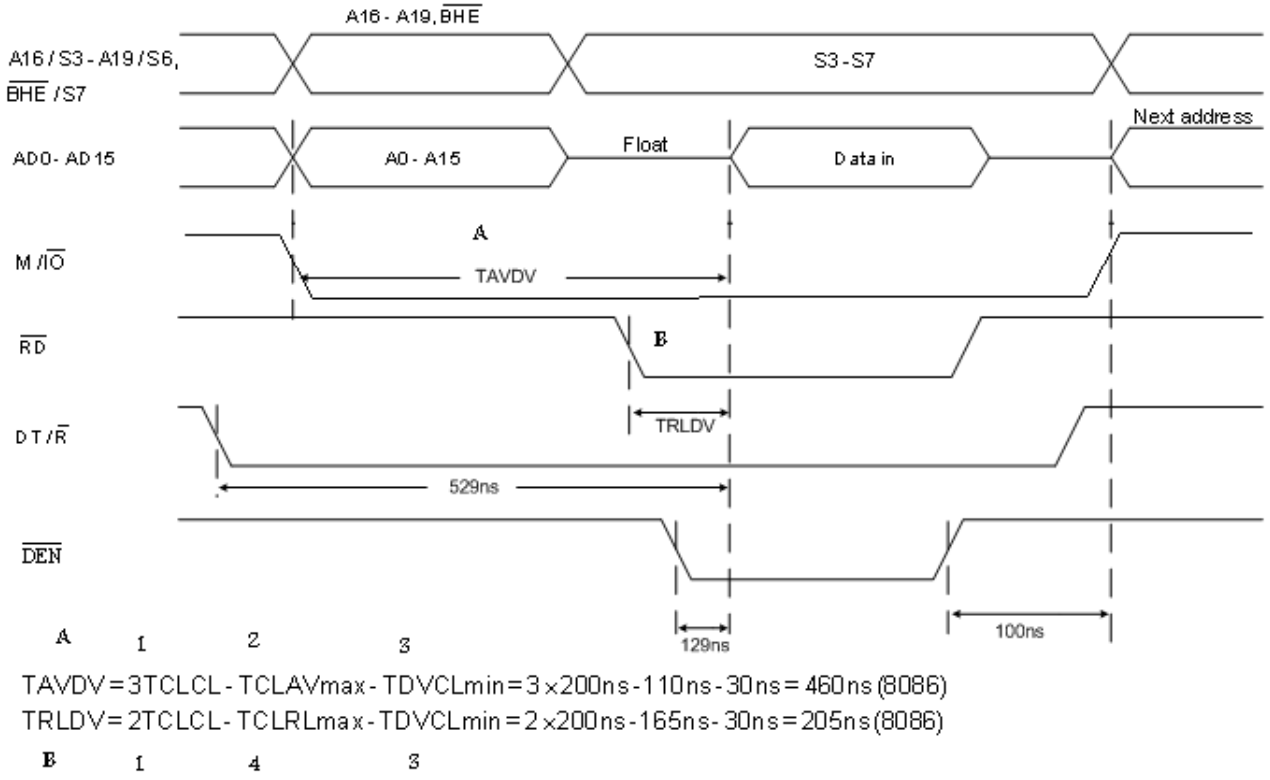
باس کنترل

نوع	دستورالعمل	باس آدرس	باس داده	مد حداقل	مد حداکثر
مستقیم	IN AL (or AX), پورت	آدرس پورت = A0-A7 A8-A19=0	بایت زوج=D0-D7 بایت فرد=D8-D15 کلمه زوج=D0-D15	$M/\overline{IO}$=0 $\overline{RD}$=0	$\overline{IORC}$
	OUT پورت, AL(or AX)	آدرس پورت = A0-A7 A8-A19=0	بایت زوج=D0-D7 بایت فرد=D8-D15 کلمه زوج=D0-D15	$M/\overline{IO}$=0 $\overline{WR}$=0	$\overline{IOWC}$=0 $\overline{AIOWC}$=0
غیر مستقیم	IN AL (or AX), DX	آدرس پورت = A0-A15 A16-A19=0	مانند بالا	مانند بالا	مانند بالا
	OUT DX, AL (or AX)	آدرس پورت = A0-A15 A16-A19=0	مانند بالا	مانند بالا	مانند بالا

$\overline{BHE}$و A0 به صورت زیر کدگذاری می‌شوند:

BHE	A0	
0	0	دسترسی کلمه‌ای
0	1	دسترسی به بایت‌های زوج
1	0	دسترسی به بایت‌های فرد
1	1	بی تاثیر

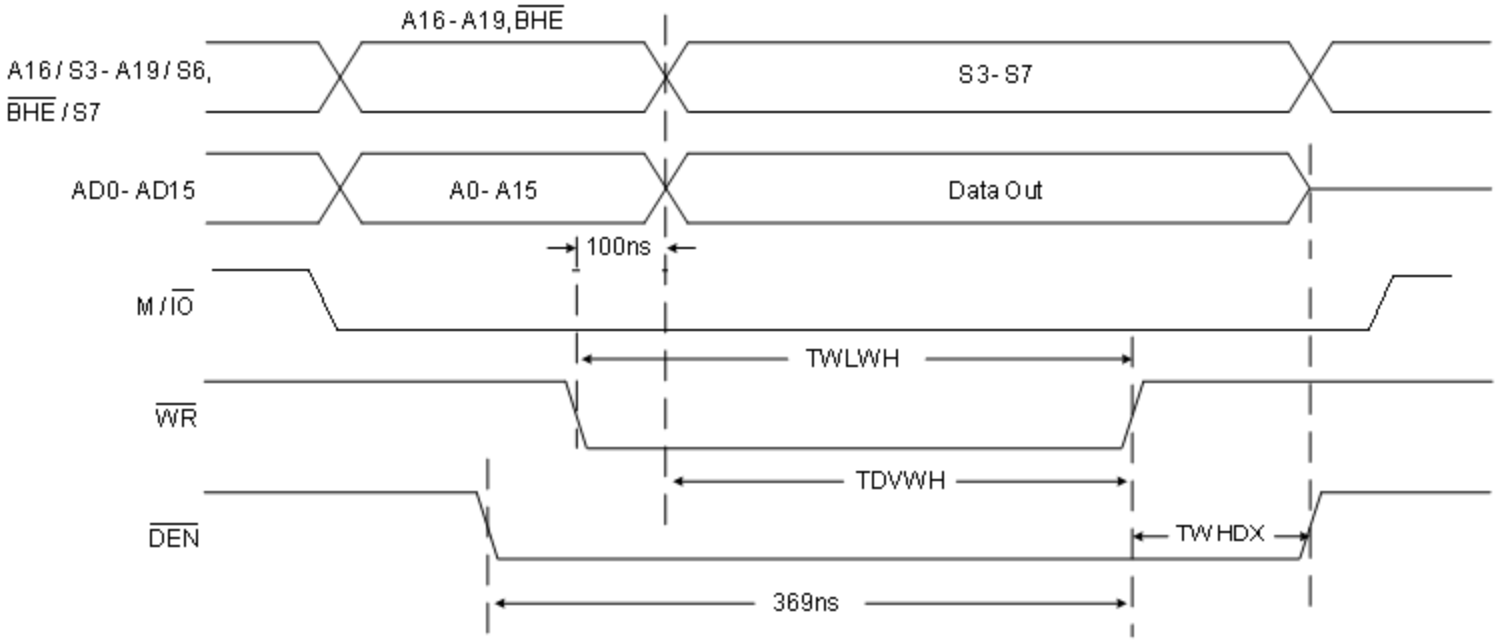
زمانبندی I/O در سیکل خواندن (مد مینیمم)



TAVDV: Address Access Time
TCLCL: Clock cycle periode
TCLAV: Address Valid Delay
TDVCL: Data Setup Time

TRLDV: Read Access Time
TCLRL:RD Active Delay

زمانبندی I/O در سیکل نوشتن (مد مینیمم)



$DT/\overline{R} = V_{OH}$

$TWLWH = 2TCLCL - 60ns = 2 \times 200ns - 60ns = 340ns$ (8086)

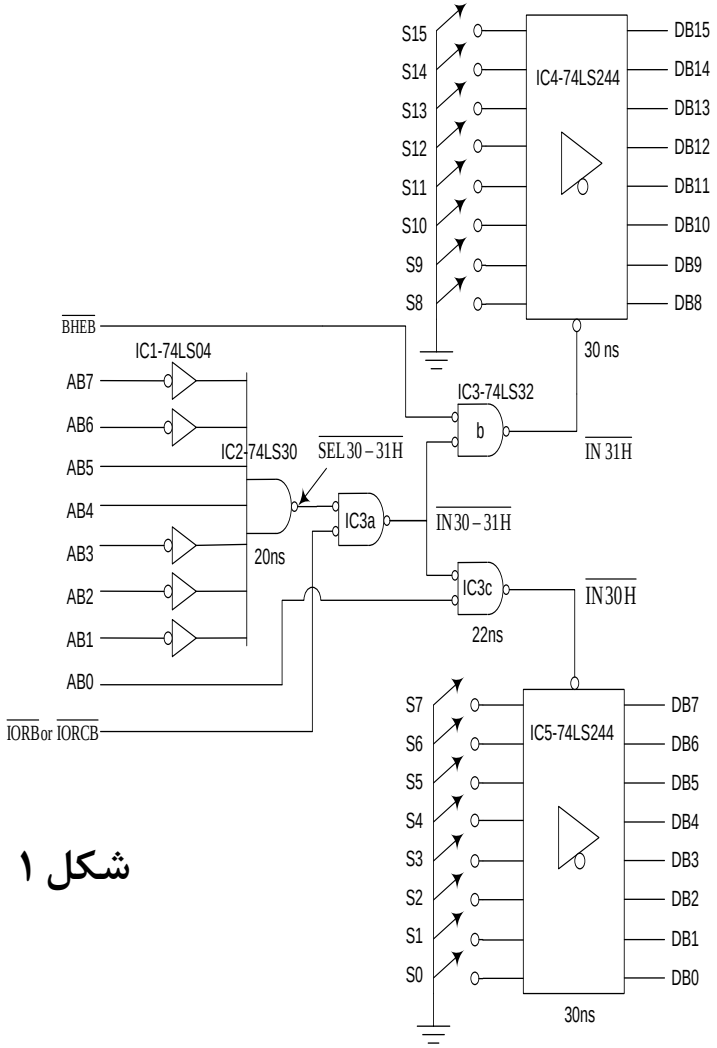
$TDVWH = 2TCLCL - TCLDV_{max} + TCVCTX_{min} = 2 \times 200ns - 110ns + 10ns = 300ns$ (8086)

$TWHDX = TCLCH - 30ns = 118ns - 30ns = 88ns$ (8086)

TCLCL: Clock cycle periode
TCLDV: Data Valid Delay
TCVCT: Control Active Dealy 1
TCLCH:Clock Low Time
TWHDX: Data Hold Time After Write

طراحی پورت ورودی موازی (ادامه)

شکل زیر یک مدار شامل دو پورت ورودی را نشان می دهد:



شکل ۱

طراحی پورت ورودی موازی

- برای شبیه‌سازی داده‌ی ورودی، از ۱۶ سوئیچ استفاده می شود. IC1 و IC2 آدرس پورت (مستقیم) قرار گرفته بر AB0 تا AB7 را دیکود می‌کنند. خروجی IC2 سیگنال انتخاب پورت است.

- تراشه‌ی IC3a سیگنال‌های $\overline{\text{SEL 30} - \text{31H}}$ و $\overline{\text{IORB}}$ را با هم ترکیب می‌کند تا پالس انتخاب دستگاه (DSP) را بسازد: $\overline{\text{IN 30} - \text{31H}}$

- چنین نامگذاری به این دلیل است که این سیگنال تنها برای دستورات ورودی که پورت 30H یا 31H را انتخاب می‌کنند فعال می‌شود.

- دستگاههای ورودی به آدرس‌های فرد و زوج تقسیم‌بندی می‌شوند. داده‌ی پورت‌های زوج از مسیر D0-D7 و پورت‌های فرد از مسیر D8-D15 انتقال می‌یابد.

- $\overline{\text{BHEB}}$ و A0 با $\overline{\text{IN 30} - \text{31H}}$ ترکیب می‌شوند تا پالس انتخاب دستگاه فرد یا زوج را به طور مجزا تولید کنند.

.

- این سیگنال‌ها نهایتا به ورودی فعال‌ساز بافرهای سه حالته وصل می‌شوند و موجب قرار گرفتن داده بر خطوط باس می‌گردند.

- DSP: Device Select Pulse

طراحی پورت ورودی موازی (ادامه)

مثال: زیربرنامه‌ای بنویسید که بازبودن حداقل یکی از سوئیچ‌های ۲، ۳، ۱۱ و ۱۳ را بررسی کند. اگر حداقل یکی از سوئیچ‌ها باز بود، با **CF=1** را برگرداند و اگر هیچ یک از کلیدها باز نبود **CF=0** شود.

حل: بازبودن سوئیچ، آن را در سطح منطقی 1 قرار می‌دهد.

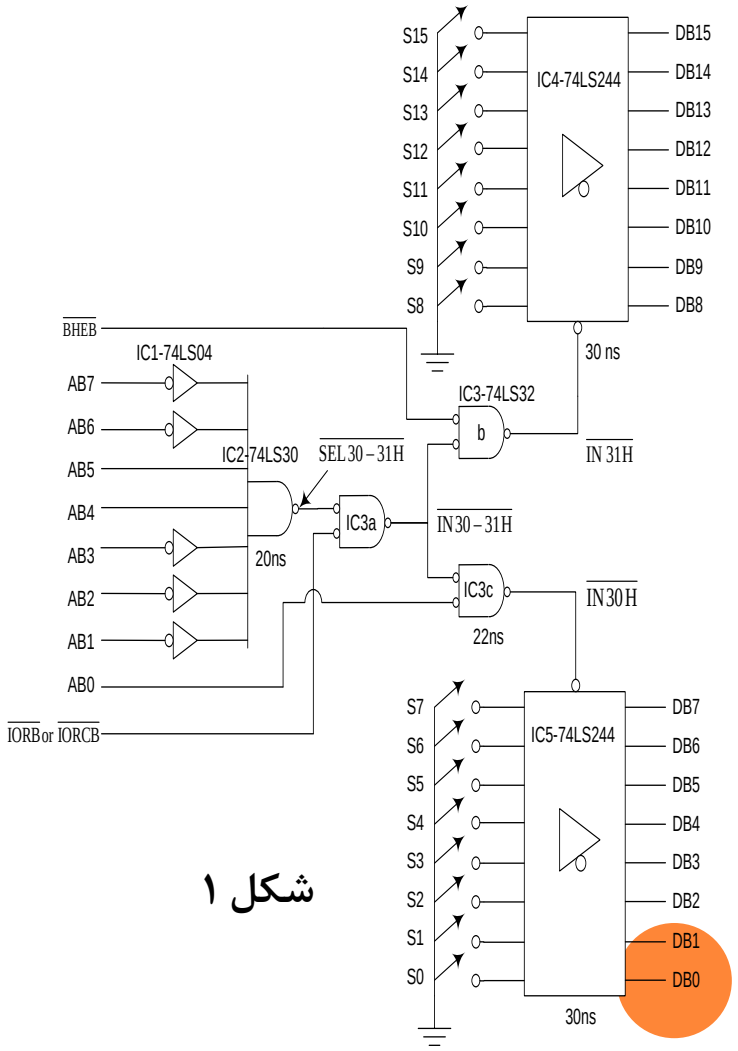
باید در برنامه 1 بودن بیت‌های خواسته شده را بررسی کنیم.

این کار با دستور **TEST AX, 280CH** انجام می‌شود که در صورت 1 بودن یکی از بیت‌های خواسته شده، نتیجه‌ی غیرصفری برمی‌گرداند.

عبارت **PUBLIC PROC1** اجازه می‌دهد که این زیربرنامه به برنامه‌های دیگر پیوند یابد.

طراحی پورت ورودی موازی (ادامه)

```
;This function test if bits 2,3,11or 13 of 16-bit data port are high or not.
;INPUT: status information from IPORT
;OUTPUT: CF=1 if condition occurred, else CF=0
;DESTROYS: AX, flags
                        PUBLIC PROC1        ;comment
                        IPORT EQU 30H       ;input port = 0030H
0000            CODE    SEGMENT BYTE PUBLIC
                        ASSUME CS:CODE
0000            PROC1   PROC NEAR
0000  F8                CLC                         ;Be sure CF=0
0001  E5 30              IN AX, IPORT                ;Sample data
0003  A9 280C            TEST AX, 0010100000001100B  ;Test
input data
0006  74 01              JZ DONE                     ;No bits high
0008  F9                 STC                ;At least one bit high
0009  C3        DONE:   RET
000A            PROC1   ENDP
000A            CODE    ENDS
                        END
```



شکل ۱

طراحی پورت ورودی موازی (ادامه)

پورت ورودی ۱۶ بیتی با هریک از دستورات زیر کار می‌کند:

```
IN AL, 30H      ; SW0-SW7 →AL
IN AL, 31H      ; SW8-SW15→AL
IN AX, 30H      ; SW0-SW15→AX
IN AL, DX       ; If DX=XX30H then SW0-SW7 →AL
                ; If DX=XX31H then SW8-SW15 →AL
IN AX, DX       ; If DX=XX30H then SW0-SW15→AX
```

به یاد داشته باشید که در دستور اول و دوم، مقصد همواره **AL** است. اگرچه داده‌ی ورودی از طریق هریک از پورت‌های **DB0-DB7** یا **DB8-DB15** ممکن است منتقل شود، واحد **BIU** به طور خودکار خط داده‌ی مناسب را انتخاب می‌کند.

طراحی پورت ورودی موازی(ادامه)

- برای دستورات دسترسی غیرمستقیم به حافظه که رجیستر DX را به کار می‌برند، آدرس I/O، ۱۶ بیتی است. در این حالت در مدار شکل ۱ بخشی از آدرس را دیکود می‌کند و لذا هر آدرسی که به 30H یا 31H ختم شود، این پورت را فعال می‌کند.

- دستور IN AX, 31H (یا IN AX, DX به فرض DX=XX31H)، یک کلمه را از پورتی با آدرس فرد و پورت با آدرس بعدی یعنی 32H می‌خواند. این کار همانند خواندن یک کلمه از حافظه در آدرس فرد است.

- تاوان این کار اضافه شدن یک سیکل باس (چهار حالت T) است.

- بنابراین مناسب‌تر است که پورت‌های ۱۶ بیتی در آدرس‌های زوج نگاشته شوند.

طراحی پورت ورودی موازی (ادامه)

مثال ۱: بررسی کنید که آیا پورت ورودی نشان داده شده در شکل ۱، زمانبندی **TAVDV** و **TRLDV** برای مد حداقل ۸۰۸۶ با کلاک **5MHz** را برآورده می‌سازد. پیکربندی کاملا بافر شده را فرض کنید.

حل: TAVDV مقدار تاخیری است که از لحظه‌ی خارج شدن آدرس معتبر از سمت **CPU** تا زمان رسیدن داده از پورت ورودی به دست **CPU**، اتفاق می‌افتد. در این مورد تاخیر ناشی ازبافرها و گیت‌هایی است که آدرس را دیکود می‌کنند.

$$T = t_{buf} + t_{IC1} + t_{IC2} + t_{IC3a} + t_{IC3b-c} = (120 + 15 + 20 + 22 + 22)ns = 199ns$$

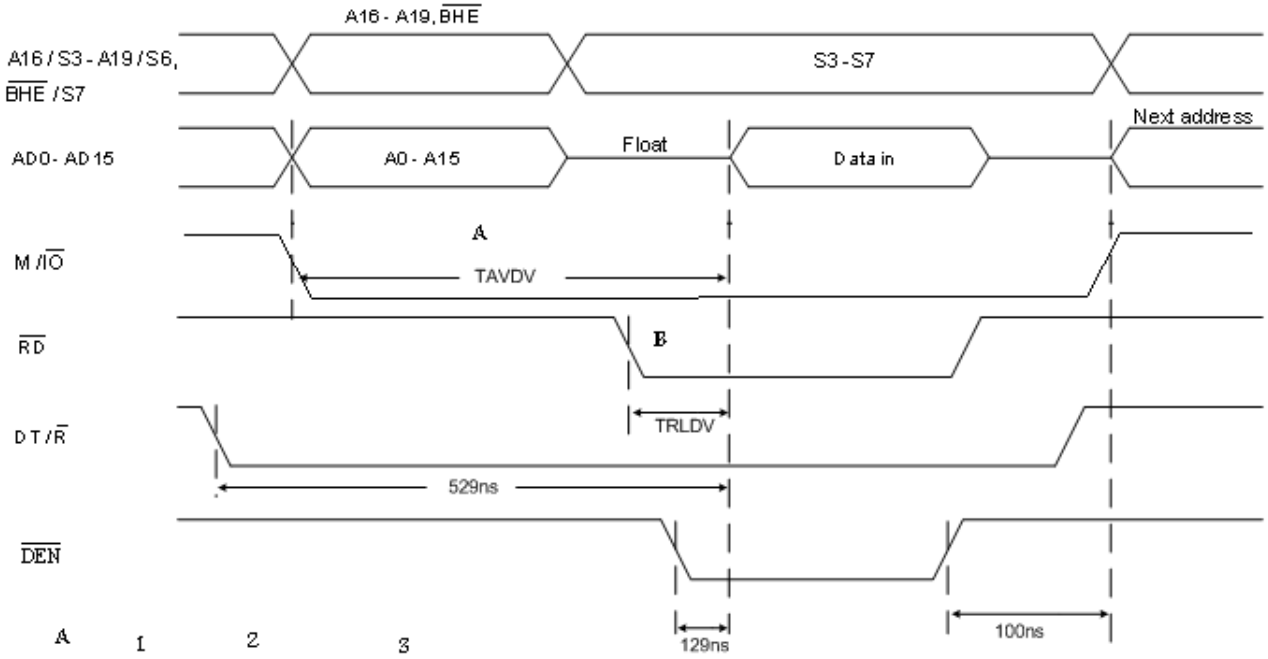
چون مقدار بالا برای ریزپردازنده ۸۰۸۶ از $460ns$ کمتر است پس قابل قبول است.

TRLDV مقدار تاخیر ایجاد شده از قرار گرفتن $\overline{RD}$ در سطح پایین تا معتبر شدن داده در CPU است. چون آدرس از قبل به خوبی در خروجی قرار گرفته است، فقط تاخیرهای زیر باید در نظر گرفته شوند:

$$T = t_{buf} + t_{IC3a} + t_{IC3b-c} = (130 + 22 + 22)ns = 174ns$$

بر طبق مشخصات، حداکثر مقدار بالا $205ns$ است.

زمانبندی I/O در سیکل خواندن (مد مینیمم)



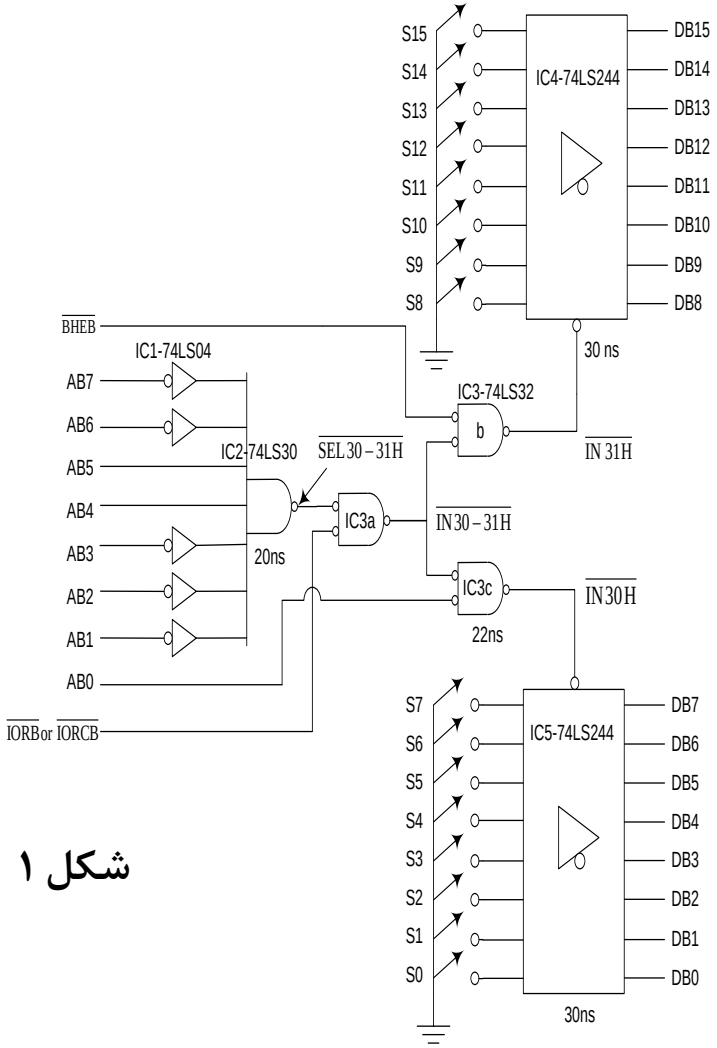
TAVDV = 3TCLCL - TCLAVmax - TDVCLmin = 3 × 200ns - 110ns - 30ns = 460ns (8086)

TRLDV = 2TCLCL - TCLRLmax - TDVCLmin = 2 × 200ns - 165ns - 30ns = 205ns (8086)

B 1 4 3

طراحی پورت ورودی موازی (ادامه)

مدار مثال ۱:



شکل ۱

طراحی پورت خروجی موازی

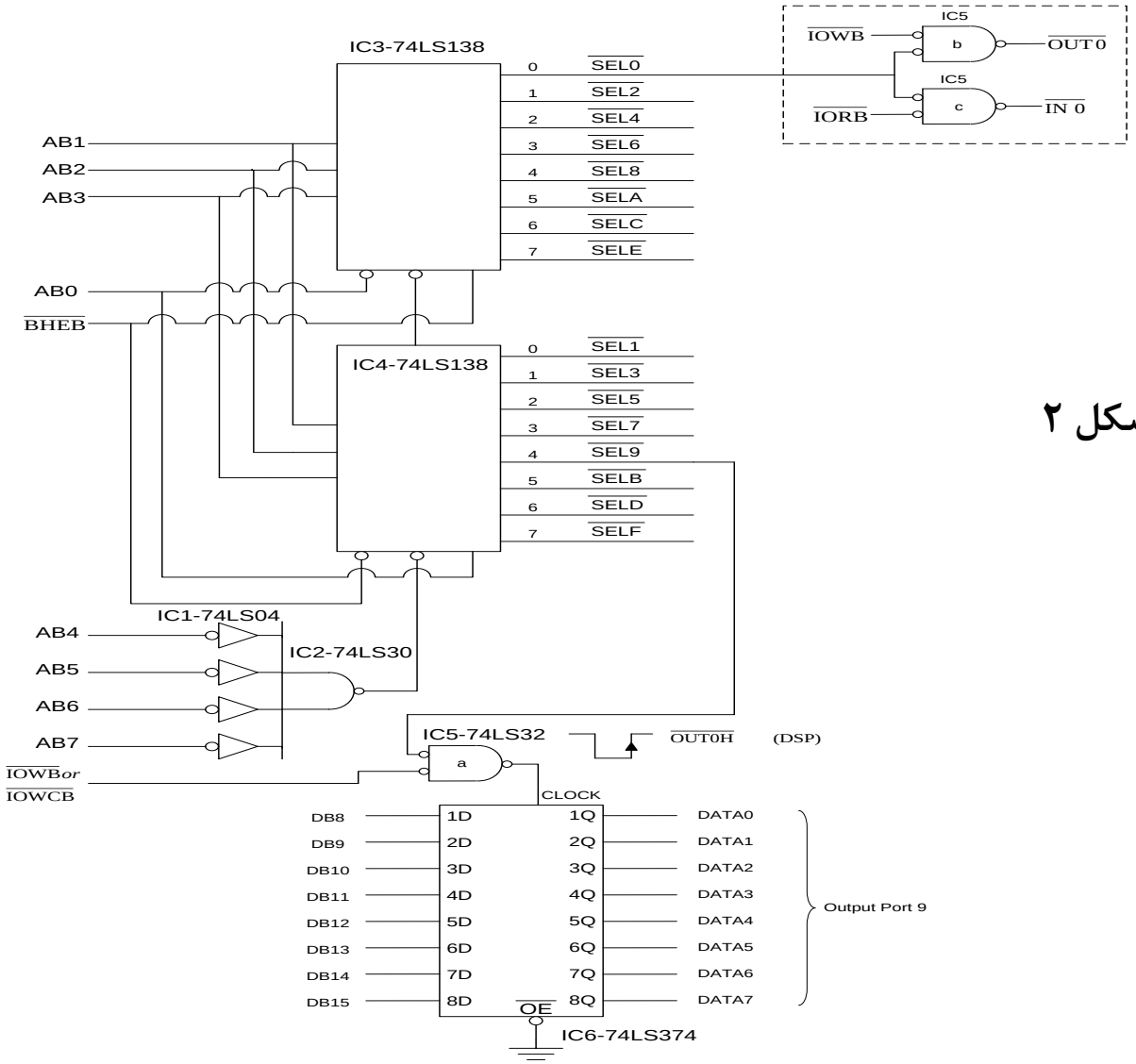
- سخت‌افزار مورد نیاز برای طراحی پورت خروجی، مشابه پورت ورودی است به جز اینکه سیگنال **Device Select Pulse (DSP)** به جای فرمان دادن به بافرهای سه حالته، به یک **لچ** فرمان می‌دهد.

- وجود لچ به این دلیل است که داده مدت زمان کوتاهی در خروجی **CPU** قرار می‌گیرد.

- شکل ۲، مداری برای فراهم کردن ۱۶ سیگنال انتخاب مجزا را فراهم می‌کند.

- AB0 و $\overline{\text{BHEB}}$ به گونه‌ای سیم‌کشی شده‌اند که IC3 سیگنال انتخاب آدرس‌های زوج و IC4 آدرس‌های فرد را ایجاد می‌کند.

طراحی پورت خروجی موازی (ادامه)



شکل ۲

ریزپردازنده ۱

محمد مهدی همایون پور

طراحی پورت خروجی موازی (ادامه)

- در این مورد خاص $\overline{SEL\ 9}$ با $\overline{IOWB}$ در **IC5a** ترکیب می‌شود و سیگنال $\overline{OUT9}$ را تولید می‌کنند.
- لبه‌ی بالا رونده‌ی این سیگنال (و در واقع $\overline{IOWB}$) باعث می‌شود که **74LS374** داده‌ی موجود بر پورت **DB8-DB15** را لچ کند.
- دو دستور **OUT 9, AL** و OUT DX, AL اگر **DX=XX09** باشد از این مدار پورت خروجی استفاده می‌کنند.

طراحی پورت خروجی موازی(ادامه)

مثال: برنامه‌ای بنویسید که در شکل ۱ بررسی کند آیا هیچ کدام از سوئیچ‌های **2**، **3**، **11** یا **13** باز هستند. اگر شرایط برقرار بود، **00** را در این پورت خروجی بنویسد. برنامه باید به طور نامحدودی تکرار شود.

حل:

می‌توان از زیربرنامه **PROC1** نوشته شده در مثال قبل برای خواندن داده از پورت ورودی استفاده کرد.

بسته به شرایط **CF**، مقادیر **00** یا **FF** در پورت خروجی نوشته می‌شود. برنامه‌ی مربوطه در زیر آمده است.

در این برنامه **PROC1** به عنوان یک روال خارجی اعلان شده است و لذا بدون نیاز به نوشتن دوباره‌ی آن، به برنامه پیوند می‌یابد.

طراحی پورت خروجی موازی (ادامه)

```
;This program calls the routine in  PROC1.
;If switches 2,3,11,13 are open FFH is output else 00.
                        EXTRN PROC1: NEAR
                        OPORT EQU 09              ;output port = 0009H
0000            CODE    SEGMENT
                        ASSUME CS: CODE
0000  B3FF      START:  MOV BL, 0FFH              ;Open switches code
0002  E8 0000 E         CALL PROC1                ;Test switches
0005  72 02              JC SET                   ;Condition met
0007  B3 00              MOV BL, 0                ;Condition not met
0009  8A C3        SET:  MOV AL, BL               ;Output code
000B   E6 09             OUT OPORT, AL            ;to OPORT
000D   EB F1             JMP START                ;Monitor continuously
000F            CODE    ENDS
                        END START
```

طراحی پورت خروجی موازی (ادامه)

مثال: در تراشه‌ی 74LS374 به کار رفته در شکل ۲، پارامترهای زمانبندی به صورت $t_{hold} = 0ns$ و $t_{setup} = 20ns$ است. بررسی کنید که این مشخصات در یک ماجول CPU کاملابافر شده با مد حداقل و کلاک 5MHz ارضاء می‌شود.

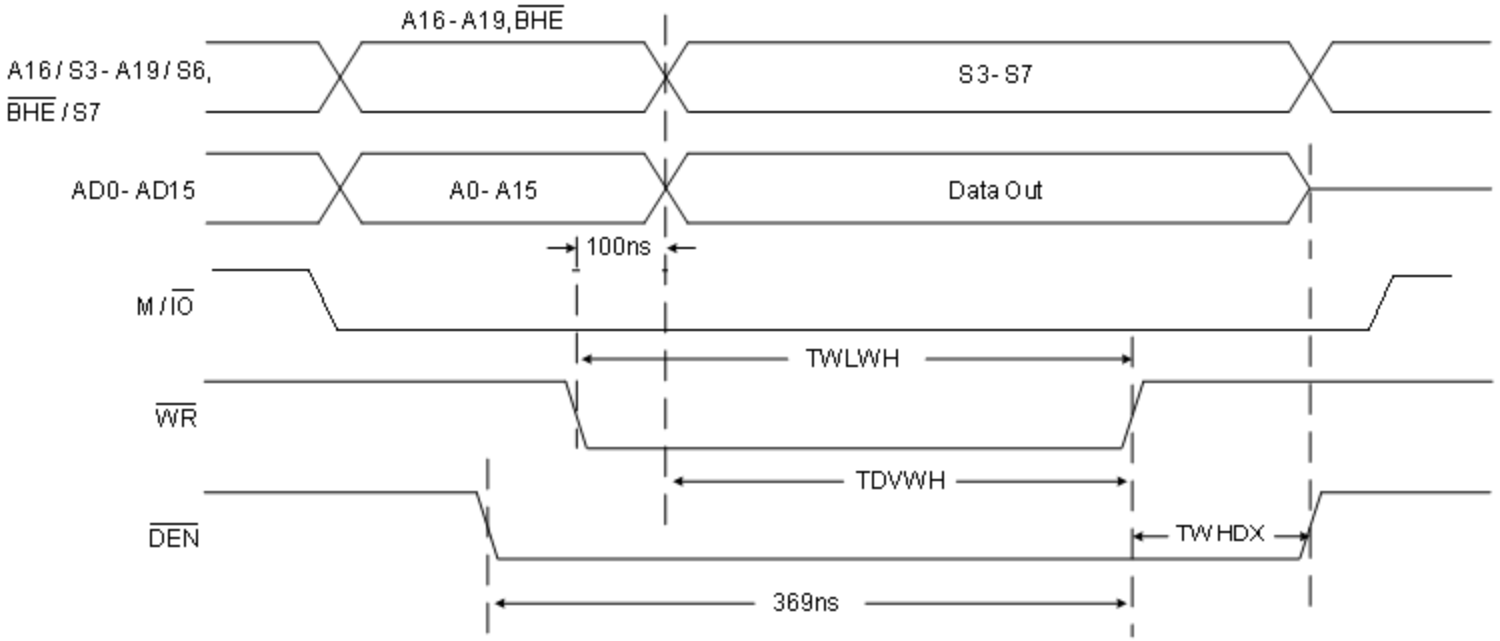
حل: 8086 داده را TDVWH نانوثانیه قبل از لبه‌ی بالا رونده‌ی $\overline{\text{IOW}}$ خارج می‌کند. داده به خاطر بافرهای باس داده، ۶۰ نانوثانیه تاخیر می‌یابد. به هرحال لبه‌پشتی $\overline{\text{IOW}}$ نیز به همان مقدار تاخیر به اضافه‌ی IC5 تاخیر می‌یابد. در بدترین حالت لازم است که

شرط نامساوی زیر برقرار باشد:

$$t_{su} \leq \text{TDVH}_{\text{min}} - (t_{DBbufmax} - t_{CBbufmin} - t_{ICbufmin})$$

در این رابطه t_{DBbuf}بیانگر تاخیر باس داده و t_{CBbuf} تاخیر بافر باس کنترل برای $\overline{\text{IOW}}$ است.

زمانبندی در سیکل نوشتن در I/O در مود مینیمم



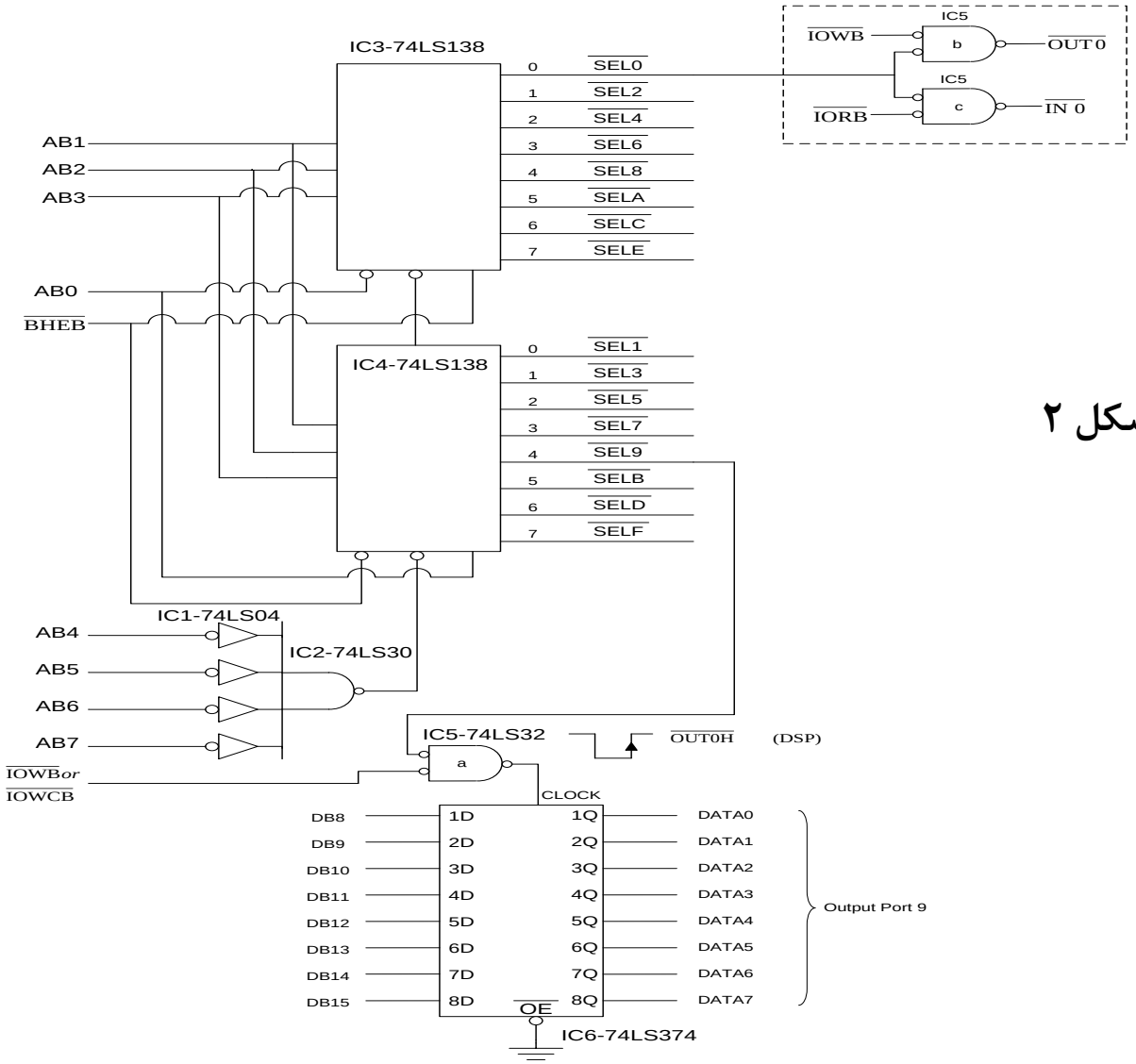
$DT/\overline{R} = V_{OH}$

$TWLWH = 2TCLCL - 60ns = 2\times 200ns - 60ns = 340ns$ (8086)

$TDVWH = 2TCLCL - TCLDV^{6}max + TCVCTX^{7}min = 2\times 200ns - 110ns + 10ns = 300ns$ (8086)

$TWHDX = TCLCH - 30ns = 118ns - 30ns = 88ns$ (8086)

طراحی پورت خروجی موازی (ادامه)



شکل ۲

طراحی پورت خروجی موازی(ادامه)

حتی با در نظر گرفتن مقدار 0 برای تاخیرهای $t_{CBbufmin}$و $t_{IC\,5\,min}$ خواهیم داشت:

$$t_{su} \leq 300 - 60 + 0 = 240ns$$

و لذا شرایط خواسته شده به راحتی تحقق می‌یابد.

با مراجعه مجدد به تصویر، داده به مدت TWHDX نانوثانیه بعد از آنکه $\overline{IOW}$ بـه وضـعیت high برگشـت، بـر خطوط باس نگه داشته می‌شود. در این مورد و در بدترین حالت خواهیم داشت:

$\text{TWHDX}_{\text{min}} = 88$

$t_{DBbufmin} = 30$, $t_{CBbufmax} = 22$, $t_{IC\,5\,max} = 0$

که بازهم به سادگی قابل تحقق است.

$t_h \leq \text{TWHDX}_{\text{min}} - (t_{CBbufmax} + t_{IC5max} - t_{DBbufmin})$

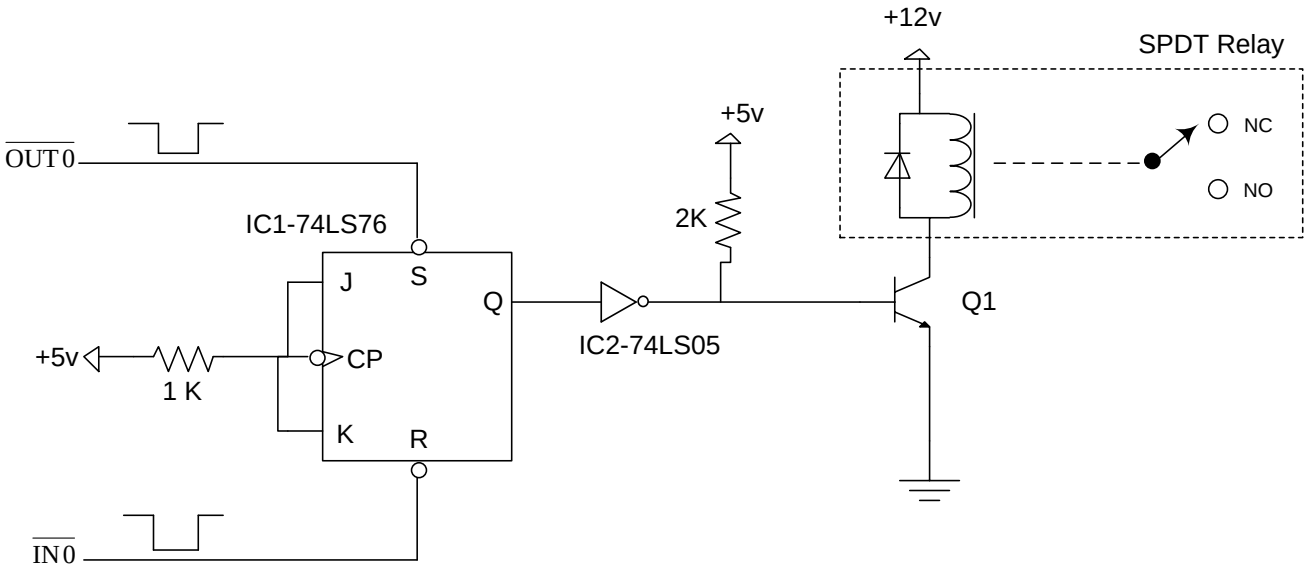
$\leq 88 - (30 + 22 - 0) = 36ns$

کاربردهای پالس انتخاب‌کننده‌ی دستگاه

معمولا پالس انتخاب‌کننده‌ی دستگاه **(DSP: Device Select Pulse)** برای فعال کردن یک لچ یا مجموعه‌ای از گیت‌های سه حالته به کار می‌رود. به هرحال گاهی نیز پالس به تنهایی مورد نیاز است.

در شکل زیر (شکل ۳)، دو سیگنال **DSP** بنام‌های $\overline{OUT\ 0}$ و $\overline{IN\ 0}$ که دو خروجی در شکل ۲ هستند، برای کنترل یک رله مکانیکی بکار گرفته شده است.

دستور **IN AL,0** فلیپ‌فلاپ را ریست می‌کند و ترانزیستور روشن می‌شود. جریان در رله برقرار می‌شود و اتصال **(NO: Normally Open)** برقرار می‌گردد. با این اتصال می‌توان هر وسیله‌ی الکتریکی را کنترل کرد.



شکل ۳

کاربردهای پالس انتخاب‌کننده‌ی دستگاه(ادامه)

- تراشه‌ی **74LS05** یک **NOT** کلکتور باز است. خروجی آن را با یک مقاومت می‌توان به **+5v** وصل کرد و بیس ترانزیستور را از طریق این مقاومت راه‌اندازی کرد.

- وجود ترانزیستور از آن جهت لازم است که جریان راه‌اندازی رله بیش از آن است که تراشه‌ی **74LS05** بتواند آنرا تامین کند.

- از دیگر کاربردهای سیگنال‌های **DSP**، تولید پالس با عرض دلخواه است.

- در مدار شکل ۳، داده‌ای که در **AL** قرار می‌گیرد در هنگام اجرای دستور خروجی، مهم نیست. یا اگر دستور ورودی به کار رود، محتوای **AL** بعد از اجرای دستور معلوم نخواهد بود.

I/O نگاشته شده در حافظه (ادامه)

I/O نگاشته شده در حافظه در سخت‌افزار همانند یک پورت I/O معمولی به نظر می‌رسد ولی چون به یک آدرس حافظه نگاشته می‌شود، در نرم‌افزار توسط دستورات خواندن/نوشتن در حافظه قابل دستیابی است.

در صورت استفاده از حالت I/O نگاشت شده در حافظه، دستوری مانند MOV BH, MEMBDS موجب می‌شود که داده‌ی موجود در پورتی که در آدرس MEMBDS از فضای حافظه نگاشت شده خوانده شده و در رجیستر BH ذخیره شود.

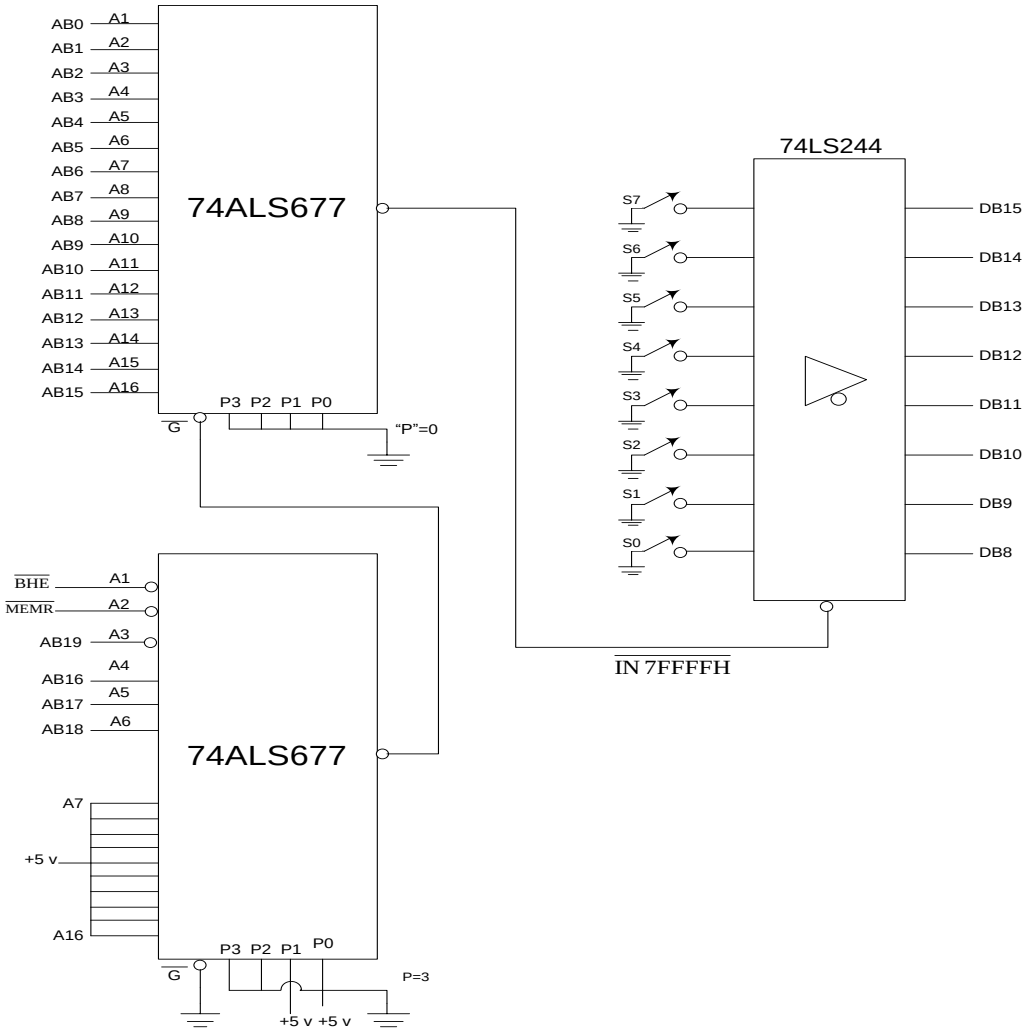
دنباله دستورات زیر اجازه می‌دهد که محتوای CX در پورت ۱۶ بیتی به آدرس DS: MEMWDS قرار گیرد:

```
LEA SI, MEMWDS;        SI به پورت اشاره می‌کند
MOV [SI], CX;          CX را بر روی پورت بفرست
```

همان‌گونه که دیده می‌شود امتیاز استفاده از I/O نگاشته شده در حافظه، به کارگیری تعداد زیادی دستور و مد آدرس‌دهی است که برای دستورات رجوع به حافظه موجود است.

I/O نگاشته شده در حافظه (ادامه)

در شکل زیر یک پورت ورودی نگاشت شده در آدرس 7FFFFH حافظه مشاهده می‌شود.



شکل ۴

I/O نگاشته شده در حافظه (ادامه)

شکل ۴ یک پورت ورودی ۸ بیتی نگاشته شده به آدرس **7FFFFH** حافظه را نشان می‌دهد. برای دیکود کردن همه‌ی ۲۰ خط آدرس و دو سیگنال کنترلی، دو تراشه‌ی **74ALS677** مورد نیاز است.

نکته اینکه دیکود کردن به گونه‌ای است که فقط این بایت با آدرس فرد قابل دستیابی است (**AB0=1**، $\overline{BHE}=0$). در اینجا یکی از عیب‌های I/O نگاشته شده **در حافظه این است که دیکود کامل آن به بررسی تمامی ۲۰ بیت خط آدرس نیاز دارد.**

دیکود جزئی برای ساده کردن دیکودر نیز قابل استفاده است ولی این کار بخشی از فضای آدرس **CPU** را غیر قابل استفاده می‌کند.

زمانی که دستگاه جانبی به بلوک بزرگی از داده‌های متوالی نیاز دارد، استفاده از I/O نگاشته شده در حافظه مناسبترین روش است. به عنوان مثالی از کاربرد I/O نگاشته شده در حافظه، یک ساعت دیجیتال را در نظر بگیرید که رجیسترهای مجزایی برای نشان دادن ساعت، دقیقه، ثانیه، روز، ماه و سال دارد.

I/O نگاشته شده در حافظه (ادامه)

شکل ۵ مدار واسط چنین تراشه‌ای را نشان می‌دهد. رجیسترهای موجود در این مدار همگی ۸ بیتی فرض شده‌اند.

هشت خط باس داده که به تراشه متصل هستند، همگی دو جهته هستند که اجازه می‌دهند با یک مجموعه خطوط I/O، دستگاه خوانده یا نوشته شود. در این مثال ۱۶ رجیستر خواندن و نوشتن وجود دارد.

۸۰۸۶ می‌خواهد بایت‌های داده با آدرس زوج را بر خطوط DB0-DB7 و آدرس فرد را بر DB8-DB15 منتقل کند.

می‌توان دستگاه جانبی را چنان دیکود کرد که همه‌ی رجیسترها در آدرس زوج (یا فرد) قرار گیرند (A0000H، A0002H، A0004H و ...) یا اینکه از مالتی‌پلکسر باس داده برای تبدیل باس ۱۶ بیتی ۸۰۸۶ به یک باس ۸ بیتی سازگار با وسیله‌ی جانبی بهره گیریم.

I/O نگاشته شده در حافظه (ادامه)

IC1 و **IC2** مالتی‌پلکسر را می‌سازند. اولی برای آدرس‌های فرد و دومی برای آدرس‌های زوج فعال می‌شود.

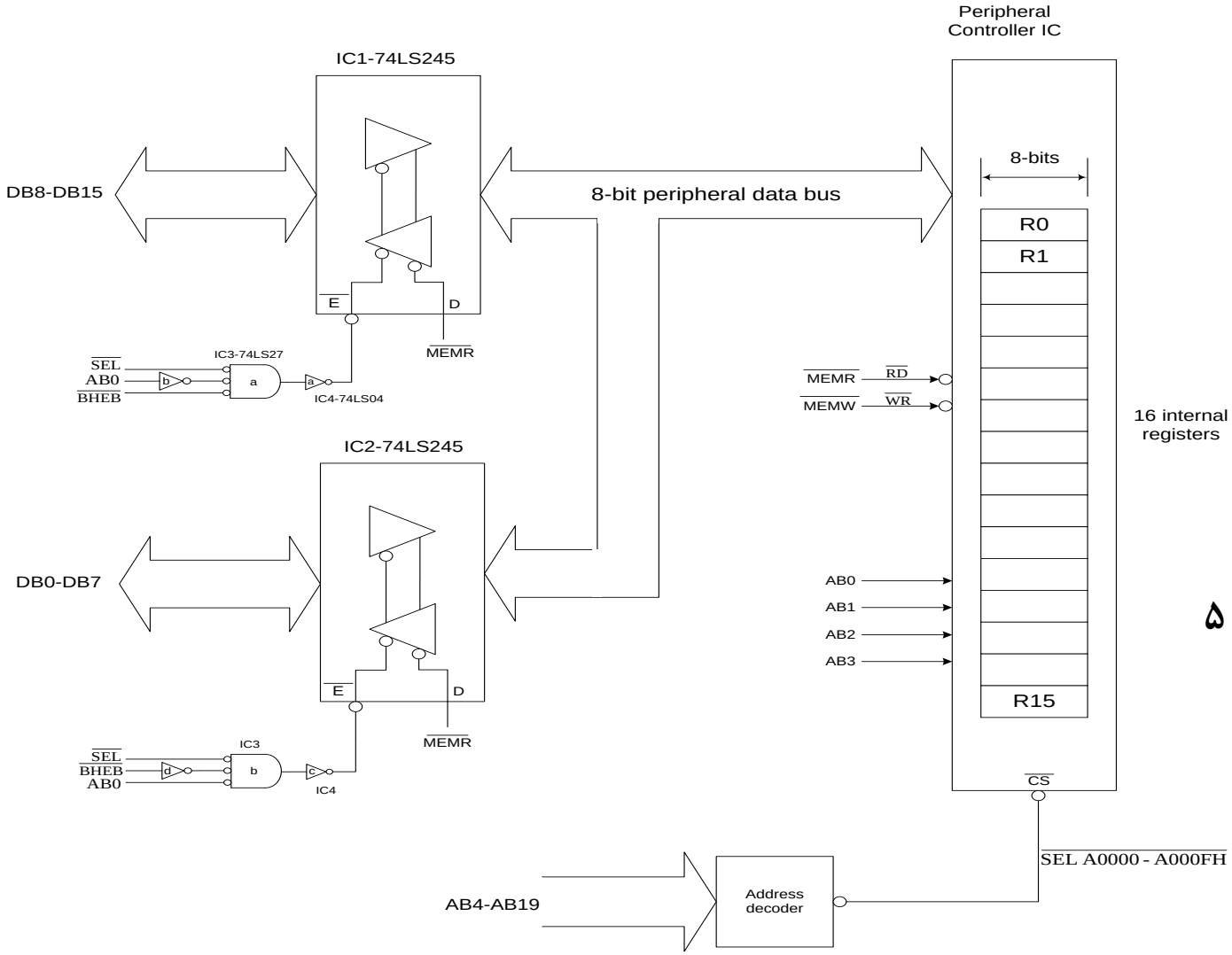
نکته اینکه اگر سیستم بخواهد دسترسی کلمه‌ای داشته باشد، هر دو بافر غیر فعال می‌شوند. سیگنال $\overline{\text{MEMR}}$ برای کنترل جهت بافرها استفاده می‌شود.

دیکودر آدرس کاملا دلخواه است و باید محدوده آدرس **A0000H-A000FH** را دیکود کند.

خروجی دیکودر آدرس، سیگنال $\overline{\text{CS}}$ تراشه کنترلر جانبی **(Peripheral Controller IC)** که شامل ۱۶ ثبات است را راه‌اندازی می‌کند.

AB0 تا **AB3** یکی از ۱۶ رجیستر را انتخاب می‌کنند. $\overline{\text{MEMR}}$ و $\overline{\text{MEMW}}$ هم خواندن یا نوشتن داده را مشخص می‌کنند.

I/O نگاشته شده در حافظه (ادامه)



شکل ۵

I/O نگاشته شده در حافظه (ادامه)

مثال: برنامه‌ای بنویسید که همه ۱۶ رجیستر کنترلر جانبی نشان داده شده در شکل ۵ را مقداردهی کند.

حل: سگمنت **PERIPH** حاوی آفست و آدرس (نگاشته شده در حافظه) تراشه‌ی کنترلر جانبی با به کارگیری نام **IO_BLOCK_ADDR** است.

بعد از واکشی آدرس سگمنت و آفست این جدول، دستور **REP MOVSB** به طور خودکار همه ۱۶ رجیستر را مقداردهی می‌کنند.

این مثال نشان دهنده انعطاف نرم‌افزاری موجود در به کارگیری پورت I/O نگاشته شده در حافظه است.

I/O نگاشته شده در حافظه (ادامه)

```
;This program initializes 16 registers of the memory-mapped peripheral controller chip.
0000                    PERIPH  SEGMENT  BYTE
0000  00 00 00 A0       IO_BLOCK_ADDR   DD   0A0000000H
                        ;Fill in the following table with the 16 bytes to be programmed

0004      10  [TABLE DB 16 DUP (?)  ?? ]
0014                    PERIPH  ENDS
0000                    CODE   SEGMENT
                        ASSUME CS: CODE, DS:PERIPH
0000                    PROC2  PROC NEAR
0000   B8  ---- R              MOV AX, PERIPH              ;Point DS:SI at the
0003   8E D8                   MOV  DS, AX                 ; programming codes.
0005   8D 36 0004 R             LEA SI, TABLE
0009   C4 3E 0000 R             LES DI, IO_BLOCK_ADDR   ;ES:DI at A000:0000.
000D   B9 0010                  MOV CX, 16                  ;16 bytes to program
0010   FC                      CLD                         ;Auto increment
0011   F3/  A4                  REP MOVSB                  ;Program the chip
0013   C3                       RET
0014                           PROC2 ENDP
0014                    CODE  ENDS
                               ENDS
```

I/O نگاشته شده در حافظه (ادامه)

- برای ریزپردازنده ۸۰۸۸، نیازی به مالتی‌پلکسر باس داده‌ی نشان داده شده در شکل ۵ نیست.
- در این حالت خطوط باس داده‌ی دستگاه جانبی را می‌توان مستقیما به خطوط باس داده‌ی سیستم وصل کرد.
- از آنجا که اکثر دستگاه‌های جانبی، ثبات‌ها و باس داده‌ی داخلی هشت بیتی را به کار می‌گیرند، استفاده از ۸۰۸۸ برای به ارتباط برقرار کردن با این دستگاه‌ها آسان‌تر است.

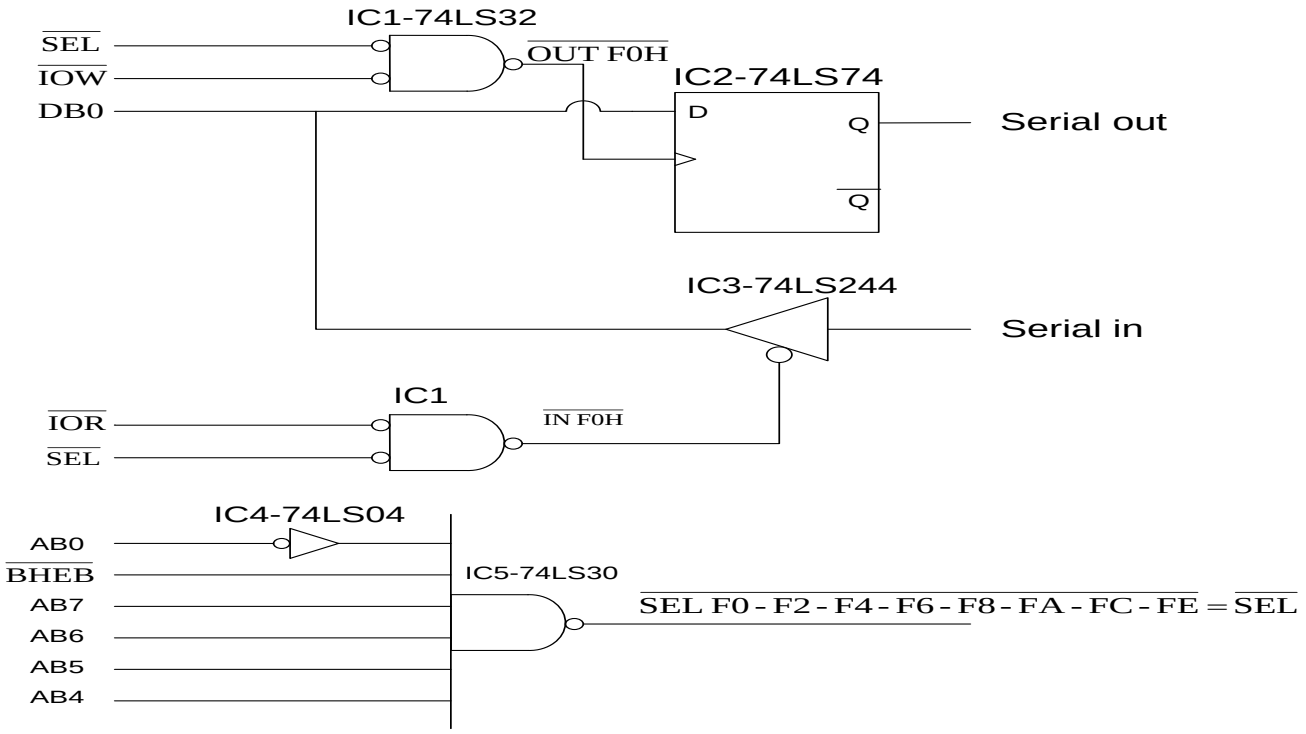
I/O سریال

یک راه نمایش داده I/O سریال، یک پورت موازی تک بیتی است مانند شکل زیر.

دیکود جزئی برای انتخاب همه‌ی پورت‌های بین XXF0H و XXFEH

سیگنال‌های انتخاب دستگاه (DSP)، داده‌ی DB0 را بر روی لچ قرار می‌دهند یـا داده‌ی سـریال ورودی را بـر روی خط باس داده‌ی DB0 قرار می‌دهند.

لذا داده‌ی سریال به عنوان بیت صفر از پورت F0H (یا F2H، F4H، ...) ارسال یا دریافت می‌شود.



شکل ۶

I/O سریال(ادامه)

- ارتباط سریال می‌تواند همگام (سنکرون) یا ناهمگام (آسنکرون) باشد.
- در ارتباط ناهمگام بیت‌های داده در یک قاب (فریم) بین بیت شروع و بیت‌های پایانی قرار می‌گیرند. در صورت تمایل می‌توان بیت توازن را نیز بعد بیت‌های داده قرار داد.
- در ارتباط ناهمگام نیازی به سیگنال ساعت برای همزمانی نیست.
- در ارتباط همگام علاوه بر خطوط ارسال یا دریافت داده، باید یک خط حاوی سیگنال ساعت نیز بین ارسال کننده و دریافت کننده برقرار نمود.

در ارتباط همگام نیازی به بیتی‌های شروع و پایان نیست ولی می‌توان بیت توازن را بعد از بیت‌های داده ارسال نمود.

بیت شروع، پایان و نرخ باود

بیت شروع و بیت‌های پایانی

- داده‌ای که می‌خواهیم بصورت همگام ارسال شود، بین یک بیت شروع (همیشه بیت **0**) و یک یـا دو بیـت پایان (همیشه بیت **1**) قرار می‌گیرد.
- بیت‌های شروع و پایان حامل داده نیستند ولی به دلیل ماهیت آسنکرون انتقال، وجود آنها لازم است.
- انتقال داده از بیت کم ارزش شروع می‌شود.

نرخ باود

- نرخ داده را می‌توان بر حسب بیت بر ثانیه یا کاراکتر بر ثانیه بیان کرد.
- اصطلاح بیت بر ثانیه را نرخ باود نیز گویند.

نرخ باودهای استاندارد برای ارتباط داده‌ی سریال
75
110
150
300
600
1200
2400
4800
9600
19200

پارامترهایی که هنگام راه‌اندازی یک پورت سریال باید تعیین شوند، عبارتند از:

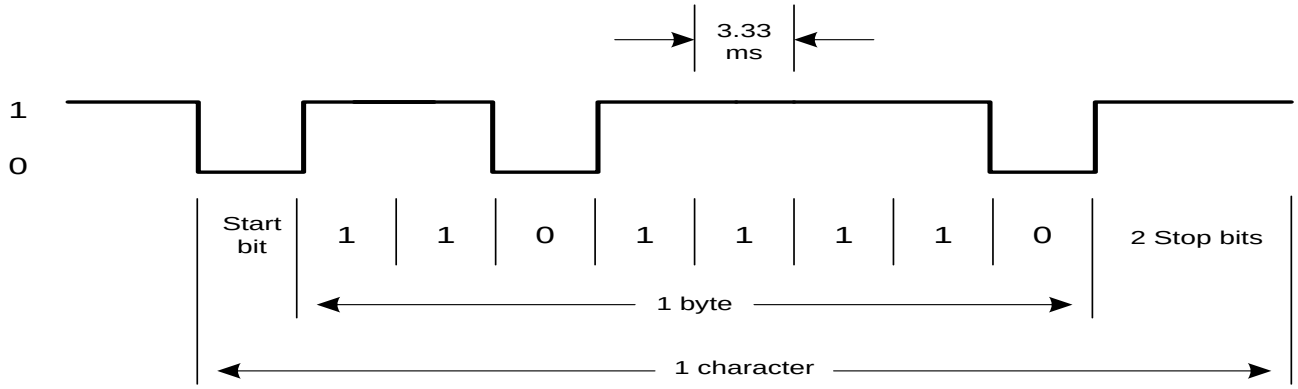
- بیت‌های داده به ازای هر کاراکتر؛ معمولا بین ۵ تا ۸.
- تعداد بیت‌های توقف، ۱ یا ۲.
- بیت توازن برای تشخیص خطای تک بیتی؛ توازن می‌تواند فرد یا زوج باشد یا اصلا به کار نرود.
- نرخ باود.

نرخ باود

مثال: نرخ باود و نرخ انتقال کاراکتر **7BH** را برای شکل زیر محاسبه کنید.

حل: چون هر بیت به مدت ۳.۳۳ میلی ثانیه موجود است، نرخ انتقال بیت برابر با ۳۰۰ =(۳.۳۳ **ms**) / ۱ بیت بـر ثانیـه یا همان ۳۰۰ باود می‌باشد.

چون به ازای هر کاراکتر ۱۱ بیت لازم است، برای انتقال هر کاراکتر به ۳۶.۶۳ میلی ثانیه زمان نیاز داریم و لذا نرخ انتقال کاراکتر برابر است با ۲۷.۳ = ۳۶.۶۳ / ۱ کاراکتر بر ثانیه.



شکل ۷

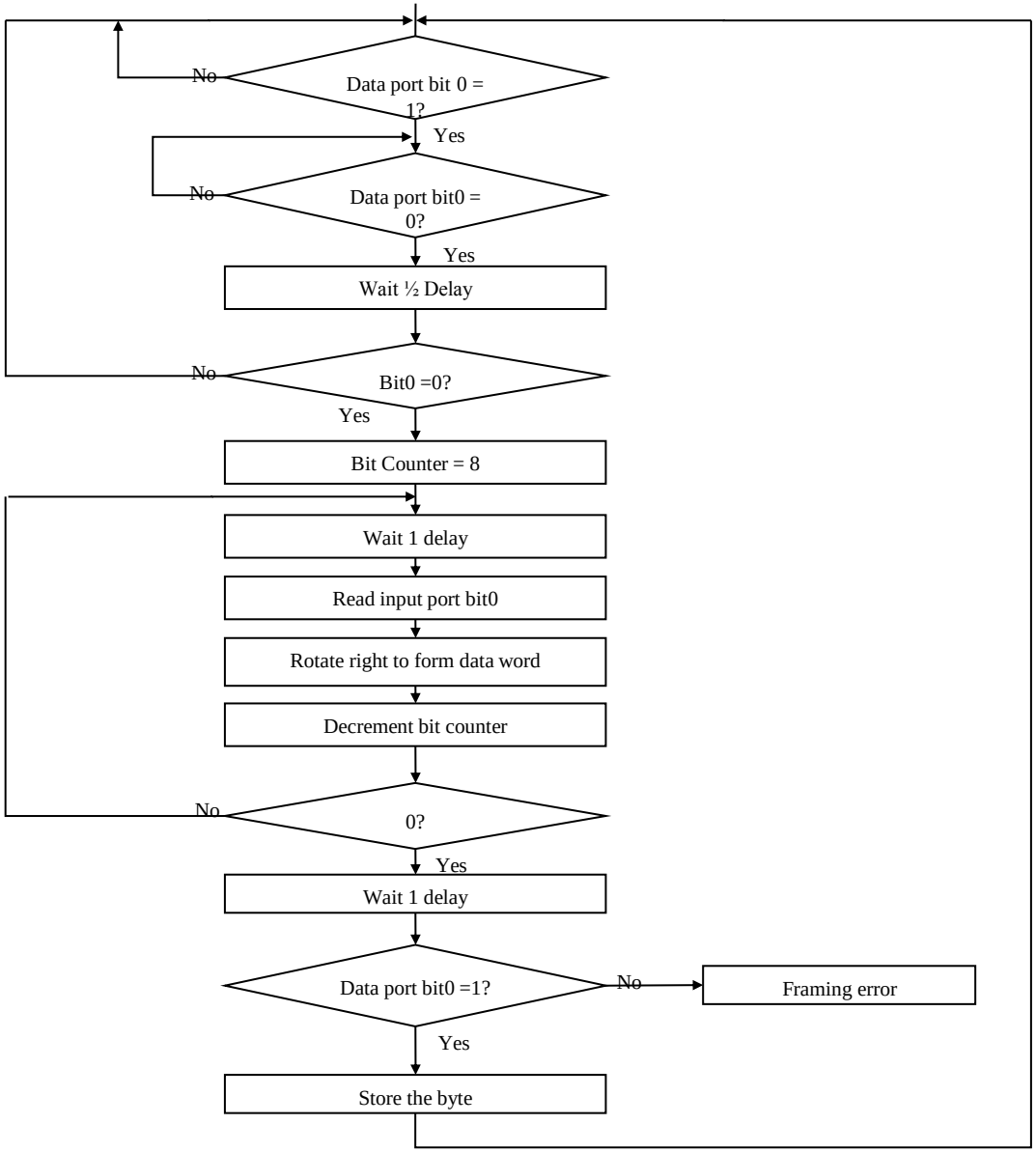
تولید داده‌ی سریال ناهمگام

مثال: برنامه زیر یک یک بایت داده را در یک قاب قرار داده و ارسال می‌نماید:

- فرض کنید بیت 0 از پورت DPORT به عنوان پین خروجی داده‌ی سریال مورد استفاده است.
- هر بیتی را که می‌خواهیم منتقل کنیم، به مکان بیت 0 از آکومولاتور شیفت می‌یابد و سپس خارج می‌شود.
- زیر برنامه‌ی DELAY نرخ باود را تعیین می‌کند.

```
                                EXTRN     DELAY:NEAR
                                DPORT     EQU 0F0H
0000                    CODE    SEGMENT   CS:CODE
                                ASSUME
0000                    PROC3   PROC      NEAR
0000  B9 000B                   MOV       CX, 11          ;11 bits/char
0003  F8                        CLC                       ;Start bit
0004  D0 D0                     RCL       AL, 1           ;Move to position 0
0006  E6 F0               TRANS: OUT       DPORT, AL       ;Transmit bit
0008  E8 0000 E                 CALL      DELAY           ;Wait
000B  D0 D8                     RCR       AL, 1           ;Next bit
000D  F9                        STC                       ;Stop bit
000E  E2 F6                     LOOP      TRANS           ;Do 11 times
0010  C3                        RET
0011                    PROC3   ENDP
0011                    CODE    ENDS
                                END
```

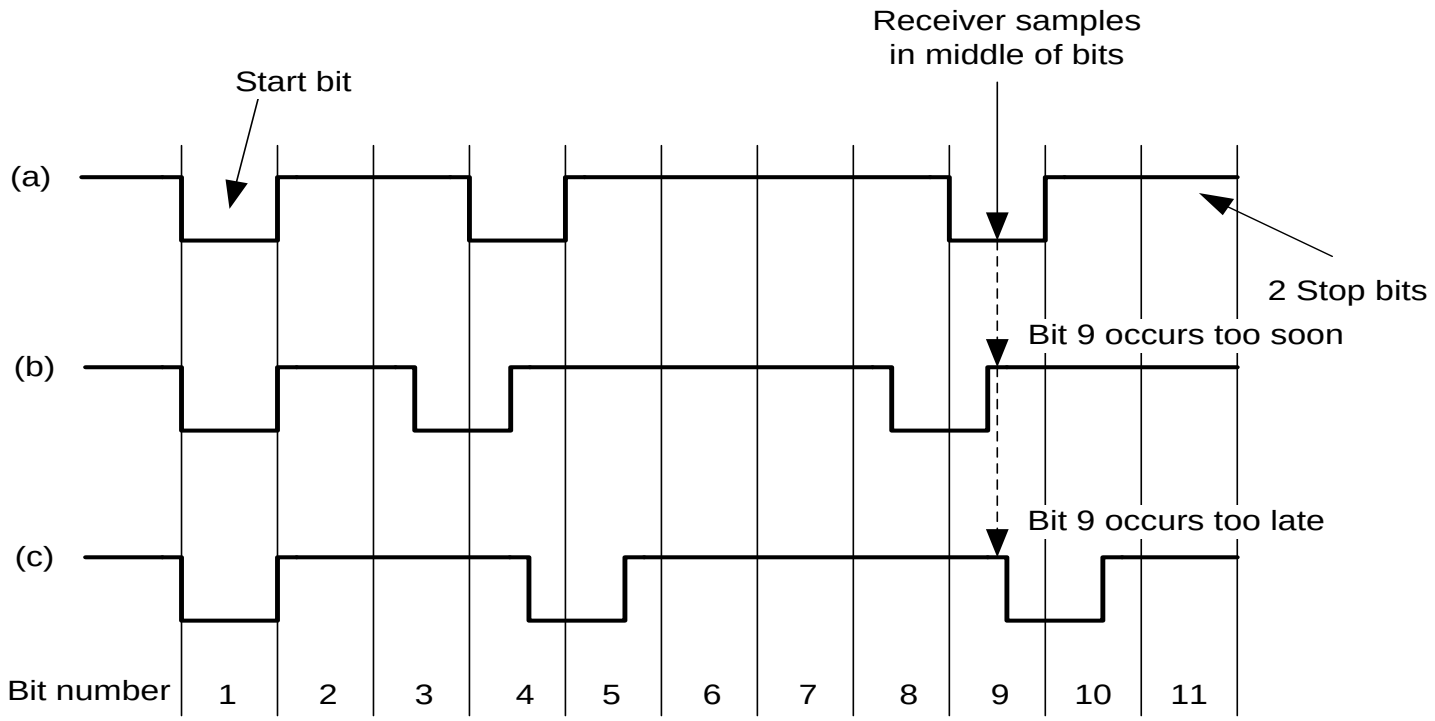
بازیابی داده‌ی سریال ناهمگام (ادامه)



فلوچارت ارتباط ناهمگام

شکل ۸

بازیابی داده‌ی سریال ناهمگام (ادامه)



شکل ۹، a) داده ی منتقل شده با نرخ مناسب، b) نرخ داده زیاد است، c) نرخ داده کم است

مدیریت ارتباط ریزپردازنده با وسایل جانبی از طریق درگاه‌ها

- در هنگام انتقال باید ریزپردازنده با سرعت دستگاه جانبی همگام شود. برخی دستگاه‌های جانبی مثل چاپگرها نمی‌توانند داده را به سرعت ریزپردازنده دریافت کنند.
- از طرف دیگر دستگاه‌های دیگری مانند دیسک‌های سخت ممکن است داده را با سرعتی بیش از ریزپردازنده درخواست کنند.
- برای جلوگیری از تلف شدن داده لازم است هر دوی این حالات به خوبی کنترل شوند.
- بدون توجه به نوع پورت I/O مورد استفاده، موازی یا سریال، یک استراتژی برای همگام‌سازی و کنترل جریان داده از طریق درگاه لازم است.

سه روش معمول کنترل و همگام نمودن جریان داده در پورت‌ها:

- I/O برنامه‌ریزی شده (Programmed I/O) با استفاده از روش سرکشی (Polling)
- وقفه (Interrupt)
- دسترسی مستقیم حافظه و وسایل جانبی به یکدیگر DAM (Direct Memory Access)

I/O برنامه‌ریزی شده

- در روش سرکشی (Pooling) میکروپروسسور مرتبا می بایست به وسیله جانبی سرکشی کند و آمادگی او برای ارسال یا دریافت داده را بررسی کند. اینکار تمام وقت **CPU** را می گیرد.
- یک چاپگر با سرعت ۱۰۰ کاراکتر بر ثانیه قادر به چاپ هر کاراکتر در مدت ۱۰ میلی‌ثانیه است.
- اما فرض کنید روتین خروجی ۸۰۸۶ برای فراهم کردن داده برای چاپگر به صورت زیر در نظر گرفته شود.

```
AGAIN:  LODSB       ;Fetch byte to AL      [12]
        OUT         DPORT   ;Send to printer       [10]
        LOOP        AGAIN   ;Do CX times           [17]
```

- اعداد نشان داده شده در براکت بیانگر تعداد کلاک‌هایی است که هر دستور به طول می‌انجامد.
- با استفاده از این روتین ارسال هر کاراکتر به ۳۹ پالس کلاک نیاز دارد.
- اگر کلاک ریزپردازنده را ۵ مگاهرتز در نظر بگیریم ۷.۸ میکروثانیه زمان نیاز داریم و لذا ۱۲۸۲۰۵ کاراکتر را می‌توان در یک ثانیه ارسال کرد که با نرخ قابل دریافت چاپگر تفاوت فاحشی دارد.

چاپگر موازی

جدول زیر سیگنال‌های چاپگر موازی معمولی را توصیف می‌کند.

یک کابل هادی با ۱۶ رشته سیم هادی برای چنین ارتباطی لازم است.

پین‌های ۱۹ تا ۳۰ به بدنه‌ی چاپگر متصل می‌شوند و برای شیلد کردن هر کدام از سیم‌های سیگنال به کار می‌روند.

پین سیگنال	پین بازگشت	سیگنال	جهت	توصیف
1	19	$\overline{STROBE}$	In	پالس STROBE برای خواندن داده‌ی ورودی. طول پالس در ترمینال دریافت باید بیش از 0.5 میکروثانیه باشد. سطح سیگنال معمولا HIGH است و خواندن داده در سطح LOW این سیگنال انجام می‌شود.
2	20	DATA 1	In	این سیگنال‌ها به ترتیب اطلاعات اولین تا هشتمین بیت داده‌ی موازی را نشان می‌دهند. هر سیگنال به ازای 1 در سطح HIGH و به ازای 0 در سطح LOW قرار می‌گیرد.
3	21	DATA 2	In	
4	22	DATA 3	In	
5	23	DATA 4	In	
6	24	DATA 5	In	
7	25	DATA 6	In	
8	26	DATA 7	In	
9	27	DATA 8	In	
10	28	$\overline{ACKNLG}$	Out	پالسی به طول تقریبی 0.5 میکروثانیه بیان می‌دارد که داده دریافت شده و چاپگر آماده‌ی دریافت داده‌ی بعدی است.

چاپگر موازی

جدول زیر سیگنال‌های چاپگر موازی معمولی را توصیف می‌کند.

یک کابل هادی با ۱۶ رشته سیم هادی برای چنین ارتباطی لازم است.

پین‌های ۱۹ تا ۳۰ به بدنه‌ی چاپگر متصل می‌شوند و برای شیلد کردن هر کدام از سیم‌های سیگنال به کار می‌روند.

پین سیگنال	پین بازگشت	سیگنال	جهت	توصیف
1	19	$\overline{STROBE}$	In	پالس STROBE برای خواندن داده‌ی ورودی. طول پالس در ترمینال دریافت باید بیش از 0.5 میکروثانیه باشد. سطح سیگنال معمولا HIGH است و خواندن داده در سطح LOW این سیگنال انجام می‌شود.
2	20	DATA 1	In	این سیگنال‌ها به ترتیب اطلاعات اولین تا هشتمین بیت داده‌ی موازی را نشان می‌دهند. هر سیگنال به ازای 1 در سطح HIGH و به ازای 0 در سطح LOW قرار می‌گیرد.
3	21	DATA 2	In	
4	22	DATA 3	In	
5	23	DATA 4	In	
6	24	DATA 5	In	
7	25	DATA 6	In	
8	26	DATA 7	In	
9	27	DATA 8	In	
10	28	$\overline{ACKNLG}$	Out	پالسی به طول تقریبی 0.5 میکروثانیه بیان می‌دارد که داده دریافت شده و چاپگر آماده‌ی دریافت داده‌ی بعدی است.

چاپگر موازی

پین سیگنال	پین بازگشت	سیگنال	جهت	توصیف
11	29	BUSY	Out	سطح HIGH بیان می‌کند که چاپگر نمی‌تواند داده‌ای دریافت کند که در موارد زیر اتفاق می‌افتد: ●در حین دریافت داده ●در حین عملیات چاپ ●در وضعیت OFF-LINE ●در وضعیت خطای چاپگر
12	30	PE	Out	سطح HIGH بیان می‌کند که کاغذ چاپگر تمام شده است.
13	-	SLCT	Out	این سیگنال بیان می‌کند که چاپگر در وضعیت خواسته شده است.
14	-	$\overline{AUTO\ FEED\ XT}$	In	قرار دادن این سیگنال در سطح LOW باعث می‌شود که بعد از چاپ به طور خودکار کاغذ یک خط به پایین رود.
15	-	NC	-	بدون استفاده
16	-	0V	-	سطح منطقی زمین
17	-	CHASIS-GND	-	زمین بدنه‌ی چاپگر که مستقل از سطح منطقی زمین است.
18	-	NC	-	بدون استفاده
19تا30	-	GND	-	سیگنال بازگشت زوج‌های به هم تابیده شده که در سطح LOW قرار می‌گیرند.

چاپگر موازی

پین سیگنال	پین بازگشت	سیگنال	جهت	توصیف
31	-	$\overline{INIT}$	In	قرار گرفتن این سیگنال در سطح LOW کنترلر چاپگر را به وضعیت اولیه‌اش بازنشانی می‌کند و بافر آن را پاک می‌کند. عرض این پالس در گیرنده باید بیش از 50 میکروثانیه باشد.
32	-	$\overline{ERROR}$	Out	در وضعیت‌های زیر سطح این سیگنال LOW می‌شود: • وضعیت اتمام کاغذ • وضعیت OFF-LINE • وضعیت خطا
33	-	GND	-	همانند پین‌های 19 تا 30
34	-	NC	-	بدون استفاده
35	-			از طریق مقاومت 4.7KΩ به 5v+ متصل می‌شود.
36	-	$\overline{SLCT\ IN}$	In	ورود داده به چاپگر تنها زمانی امکان‌پذیر است که این سیگنال در سطح LOW قرار گیرد.

چاپگر موازی

- هشت سیم داده به نام‌های DATA1 تا DATA8 وجود دارند. وقتی سیگنال $\overline{STROBE}$ به مدت 0.5 میکروثانیه یا بیشتر در سطح LOW قرار گیرد، چاپگر داده موجود بر پین‌های دیتا را لچ می‌کند.
- دو سیگنال کنترلی دیگر نیز برای چاپگر فراهم شده‌اند که BUSY و $\overline{ACKNLG}$ نام‌گذاری شده‌اند. با توجه به جدول BUSY یک سیگنال ACTIVE-HIGH است که بیان می‌کند چاپگر مشغول چاپ یک کاراکتر است، خطایی پیش آمده است یا در وضعیت OFF-LINE قرار دارد.
- $\overline{ACKNLG}$ یک پالس ACTIVE-LOW است که بعد از دریافت و چاپ یک کاراکتر از طرف چاپگر اعمال می‌شود. به تفاوت این دو سیگنال توجه کنید که BUSY یک سیگنال حساس به سطح است در حالیکه $\overline{ACKNLG}$ حساس به لبه است.
- سیگنال‌های BUSY، $\overline{ACKNLG}$ و $\overline{STROBE}$ مجموعه‌ای از سیگنال‌های Handshaking را بین چاپگر و CPU تشکیل می‌دهند. CPU دست خود را از طریق سیگنال $\overline{STROBE}$ دراز می‌کند و می‌گوید "داده حاضر است"؛ چاپگر با سیگنال $\overline{ACKNLG}$ به او پاسخ می‌دهد و می‌گوید "آن را دریافت کردم، می‌توانی داده‌ی بعدی را برایم بفرستی".
- BUSY همانند $\overline{ACKNLG}$ است ولی به جای یک پالس، یک سطح را فراهم می‌کند.

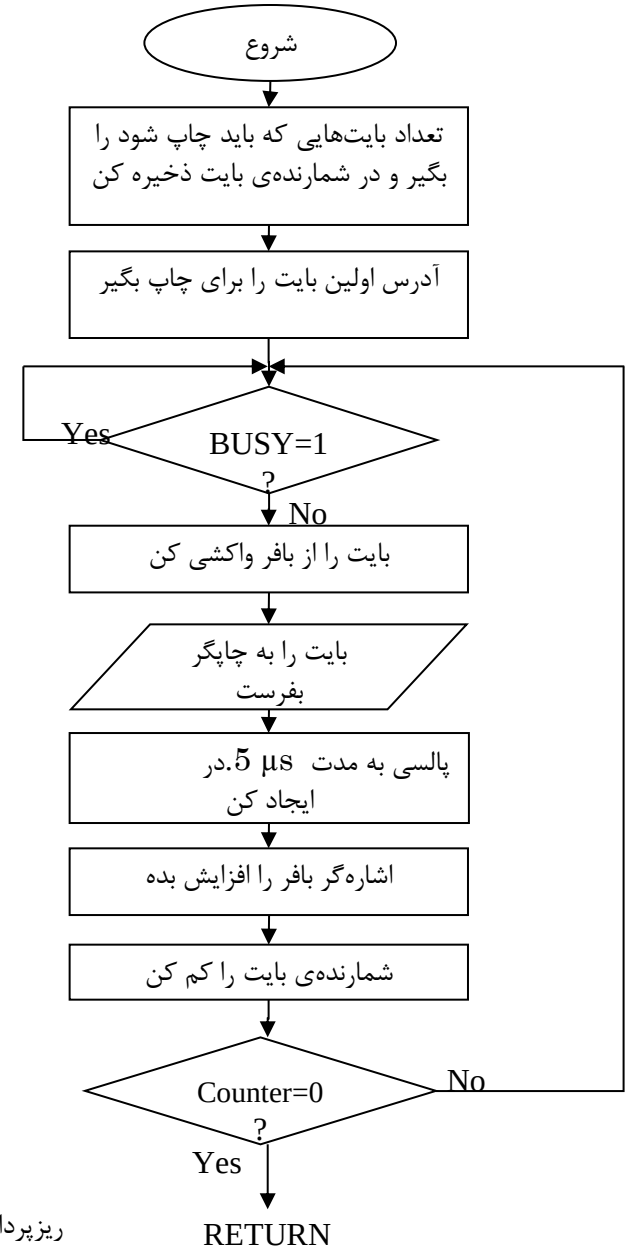
روش سرکشی (ادامه)

- شکل ۱۳ فرآیند لازم برای انتقال داده از کامپیوتر به چاپگر را نشان می‌دهد. در این برنامه فرض شده که داده‌های آماده‌ی چاپ در خانه‌های متوالی حافظه با عنوان بافر حافظه قرار گرفته‌اند. اشاره‌گری به ابتدای این بافر اشاره می‌کند و شمارنده‌ای تعداد بایت‌هایی که باید برای چاپ فرستاده شوند را ذخیره می‌کند.

- بلوک تصمیم گیری "BUSY = 1?" یک حلقه‌ی سرکشی را تشکیل می‌دهد که در آن CPU مرتبا پرچم BUSY چاپگر را بررسی می‌کند و زمانی که در سطح LOW قرار گرفت CPU داده‌ی بعدی را آماده کرده و به چاپگر می‌فرستد و اگر داده‌های دیگری همچنان باقی مانده باشد مجددا در حلقه‌ی سرکشی قرار می‌گیرد.

- قبل از نوشتن برنامه‌ای که عملیات تشریح شده را انجام دهد، سخت افزار لازم برای این ارتباط را فراهم می‌کنیم. شکل ۱۴ مدار مربوطه را نشان می‌دهد. یک لچ هشت بیتی برای نگه‌داشتن داده در پین‌های ورودی چاپگر به کار رفته است. کلاک این لچ از سیگنال DSP خروجی IC2-a می‌آید. سیگنال‌های SEL2 و SEL3 از خروجی‌های دیکودر نشان داده شده در شکل ۲ می‌آیند.

روش سرکشی (ادامه)



شکل ۱۳

فلوچارت ارسال داده برای پرینتز همراه با handshaking

روش سرکشی (ادامه)

- دنباله‌ی دستورات زیر برای تولید پالس $\overline{\text{STROBE}}$ قابل استفاده است (با این فرض که فلیپ فلاپ ابتدائا سِت شده باشد):

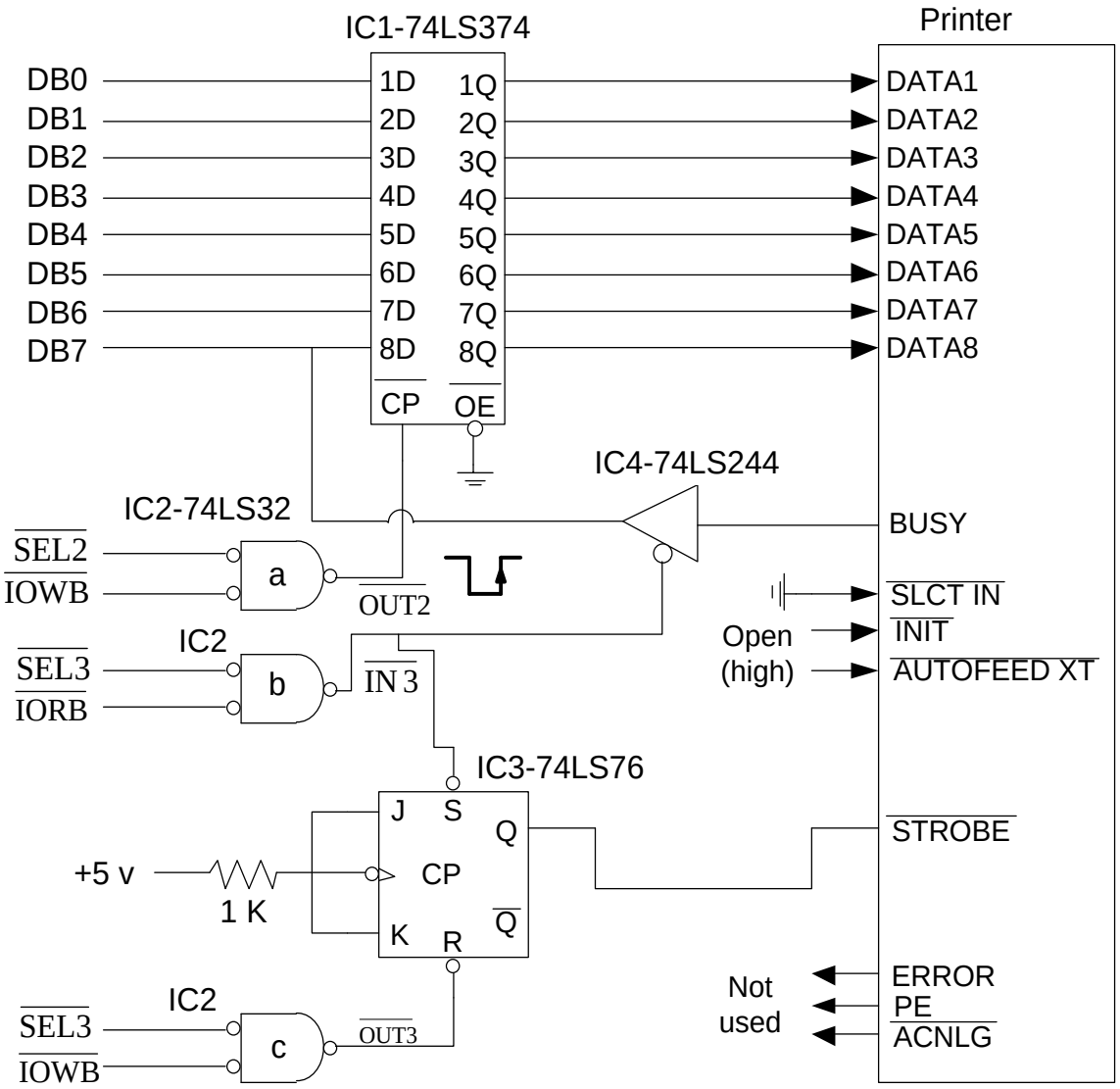
```
OUT 3, AL      ; STROBE = 0
IN AL, 3       ;
```

- چون دستور IN برای اجرا به ۱۰ کلاک نیاز دارد، با فرض کلاک 5MHz، سیگنال $\overline{\text{STROBE}}$ به مدت 2µs در وضعیت LOW قرار می‌گیرد و این مقدار چهار برابر زمان مورد نیاز است.

- نکته اینکه دستور IN AL, 3 وضعیت سیگنال BUSY را به عنوان بیت هفتم از پورت ورودی 3 می‌خواند.

- دستور IN AL, BUSY پالس $\overline{\text{STROBE}}$ را غیرفعالی می‌کند. چون دستور LOOP POLL بین دستورات فعال کردن و غیرفعال کردن $\overline{\text{STROBE}}$ اتفاق می‌افتد، عرض پالس به ۲۷ کلاک یا ۵.۴ میکروثانیه می‌رسد.

روش سرکشی (ادامه)



استفاده از سیگنال‌های **BUSY** و $\overline{\text{STROBE}}$ برای همگام کردن ریزپردازنده و دستگاه جانبی است. چون انتقال داده به چاپگر تحت کنترل نرم‌افزار انجام می‌شود، این تکنیک را **I/O برنامه‌ریزی شده** گویند.

شکل ۱۴- ارتباط چاپگر موازی با ۸۰۸۶، استفاده از I/O برنامه‌ریزی شده

روش سرکشی (ادامه)

```
;Function : Polling printer driver
;Inputs: Number of bytes and address of first byte assumed stored in PRINT_DATA segment
;Outputs: Character to be printed at port PRINTER
;Destroys: AX, CX, SI, DS, flags.

0000                    PRINT_DATA  SEGMENT WORD
0000  ????              NUMB        DW ?          ;Number of bytes
0002  ????              ADDR        DD ?          ;Address of first byte
0006                    PRINT_DATA  ENDS
                        PRINTER     EQU  2        ;Printer port=0002
                        STATUS      EQU  3        ;Busy status port=0003
0000                    CODE        SEGMENT
                                    ASSUME  CS:CODE,
                                            DS:PRINT_DATA
                        PROC4       PROC  FAR
0000  B8 ----R          START:      MOV   AX, PRINT_DATA  ;Load DS with address
0003  8E D8                         MOV   DS, AX          ;of PRINT_DATA
0005  8B 0E 0000 R                  MOV   CX, NUMB        ;Get number of bytes
0009  C5 36 0002 R                  LDS   SI, ADDR        ;Get address of data to DS:SI
000D  FC                            CLD                   ;Auto increment
000E  E4 03             POLL:       IN    AL, STATUS      ;Set STROBE=1     +[10] Input BUSY
                                                          flag
0010  A8 80                         TEST  AL, 10000000B   ;Test BUSY        +[4]
0012  75 FA                         JNZ   POLL            ;Wait until ready +[16]
0014  AC                            LODSB                 ;Get byte         [12] and advance
                                                          printer
0015  E6 02                         OUT   PRINTER, AL     ;Output to printer [10]
0017  E6 03                         OUT   3, AL           ;STROBE=0   [10]
0019  E2 F3                         LOOP  POLL            ;Do CX times   [17]
001B  E4 03                         IN    AL, STATUS      ;Quit with STROBE=1
001D  CB                            RET                   ;Then return
001E                    PROC4       ENDP
001E                    CODE        ENDS
                                    END   START
```

روش سرکشی (ادامه)

- در برنامه‌ی نوشته شده، سه دستور با علامت "+" مشخص شده‌اند که حلقه‌ی سرکشی را تشکیل می‌دهند و اعداد نوشته شده در براکت بیانگر تعداد کلاک‌های لازم برای اجرای این دستورات است که جمعا ۳۰ کلاک (6µs برای کلاک 5MHz) برای حلقه‌ی سرکشی مورد نیاز است.

- از آنجایی که چاپگر به مدت زمان بیشتری برای انجام چاپ نیاز دارد این حلقه بارها تکرار می‌شود تا چاپگر آماده‌ی دریافت کاراکتر بعدی شود. در این مثال CPU باید حلقه را 10ms/6µs=1666 بار تکرار کند.

- حلقه‌ی سرکشی راندمان پایینی دارد چون CPU مدت زمان زیادی را باید منتظر چاپگر باشد تا آماده‌ی دریافت کاراکتر بعدی شود در حالیکه تحویل دادن هر کاراکتر به آن بیش از 10µs طول نمی‌کشد.

- در این مدت CPU دستور دیگری جز سرکشی بیت وضعیت چاپگر را انجام نمی‌دهد. اگر CPUکار دیگری جز چاپ این کاراکترها نداشته باشد، این مساله اهمیتی ندارد ولی اگر بخواهیم CPU را در یک سیستم Multi tasking راه‌اندازی کنیم که در آن CPU چندین کار را با هم انجام می‌دهد، آنگاه روش سرکشی برای کار با دستگاه‌های جانبی کندی چون چاپگر مناسب نیست.

نرخ ارسال داده

- ۸۰۸۶ می‌تواند با سرعت خیلی بیشتری از آنچه در مثال چاپگر مشاهده شد، داده را ارسال کند.

- برای محاسبه‌ی این نرخ حداکثر فرض می‌کنیم دستگاه جانبی بعد از یک بار اجرای حلقه‌ی سرکشی آماده‌ی دریافت داده‌ی بعدی است.

- در این مثال ۳۰ کلاک برای بررسی پرچم BUSY، ۴۹ کلاک برای واکشی داده، افزایش اشاره‌گر، تولید پالس STROBE و تست کردن آن است (مجموعا ۷۹ کلاک) که در یک سیستم با کلاک 5MHz به 15.8μs زمان نیاز دارد و لذا نرخ ارسال داده ۶۳۲۹۱ کاراکتر بر ثانیه خواهد بود.

زمان پاسخ‌دهی

- یک سیستم مبتنی بر ریزپردازنده نوعا چندین دستگاه جانبی دارد که شامل فلاپی دیسک، چاپگر، مودم، ترمینال نمایشگر ویدئو و ... است.
- استفاده از روش سرکشی لازم می‌دارد که برای هر کدام از این دستگاه‌ها یک سیگنال BUSY/READY موجود باشد.
- شکل ۱۵ سیگنال‌های BUSY/READY برای هشت دستگاه جانبی را نشان می‌دهد که از طریق یک پورت هشت بیتی خوانده می‌شود.

7	6	5	4	3	2	1	0
Video Terminal	**Modem**	**DAC**	**ADC**	**Plotter**	**Daisy Wheel Printer**	**Line Printer**	**Floppy disk**

شکل ۱۵– سیگنال‌های BUSY/READY برای هشت دستگاه جانبی

- زمانیکه چندین دستگاه جانبی به ریزپردازنده متصل می‌شوند زمان پاسخ‌گویی به آنها مهم می‌شود.
- **زمان پاسخ‌گویی** از لحظه‌ای است که BUSY/READY=READY می‌شود تا زمانیکه CPU به آن سرویس‌دهی می‌کند. در مثال قبل چون روتین حلقه‌ی سرکشی 6µs طول می‌کشید، حداکثر زمان پاسخ‌گویی برابر با همین مقدار است.

زمان پاسخ‌دهی (ادامه)

```
POLL    IN      AL, 2           ;Read status port       [10]
        TEST    AL, 00000001B   ;Test bit 0             [4]
        JZ      SKIP0           ;Not ready so skip      [4/16]
        CALL    FD              ;Floppy disk            [15]
SKIP0   TEST    AL, 00000010B   ;Test bit 1             [4]
        JZ      SKIP1           ;Not ready so skip      [4/16]
        CALL    LP              ;Line printer           [15]
SKIP1   TEST    AL, 00000100B   ;Test bit 2             [4]
        JZ      SKIP2           ;Not ready so skip      [4/16]
        CALL    DWP             ;Daisy wheel printer    [15]
SKIP2   TEST    AL, 00001000B   ;Test bit 3             [4]
        JZ      SKIP3           ;Not ready so skip      [4/16]
        CALL    PLOT            ;Plotter                [15]
SKIP3   TEST    AL, 00010000B   ;Test bit 4             [4]
        JZ      SKIP4           ;Not ready so skip      [4/16]
        CALL    ADC             ;Analog digital conv    [15]
SKIP4   TEST    AL, 00100000B   ;Test bit 5             [4]
        JZ      SKIP5           ;Not ready so skip      [4/16]
        CALL    DAC             ;Digital analog conv    [15]
SKIP5   TEST    AL, 01000000B   ;Test bit 6             [4]
        JZ      SKIP6           ;Not ready so skip      [4/16]
        CALL    MOD             Modem                   [15]
SKIP6   TEST    AL, 10000000B   ;Test bit 7             [4]
        JZ      POLL            ;Not ready so skip      [4/16]
        CALL    TERM            ;Terminal               [15]
        JMP     POLL
```

زمان پاسخدهی (ادامه)

- مشکل واقعی خود روش سرکشی است.

- اگرچه پیاده‌سازی سخت‌افزاری و نرم‌افزاری آن ساده **است ولی راندمان پایینی در استفاده از منابع** کامپیوتری دارد.

- زمانیکه چندین دستگاه جانبی وجود دارد این روش ممکن است کاملا نارضایت بخش باشد. روش‌های بر پایه‌ی **DMA و وقفه** جایگزین مناسب روش سرکشی هستند.

I/O وقفه‌گرا

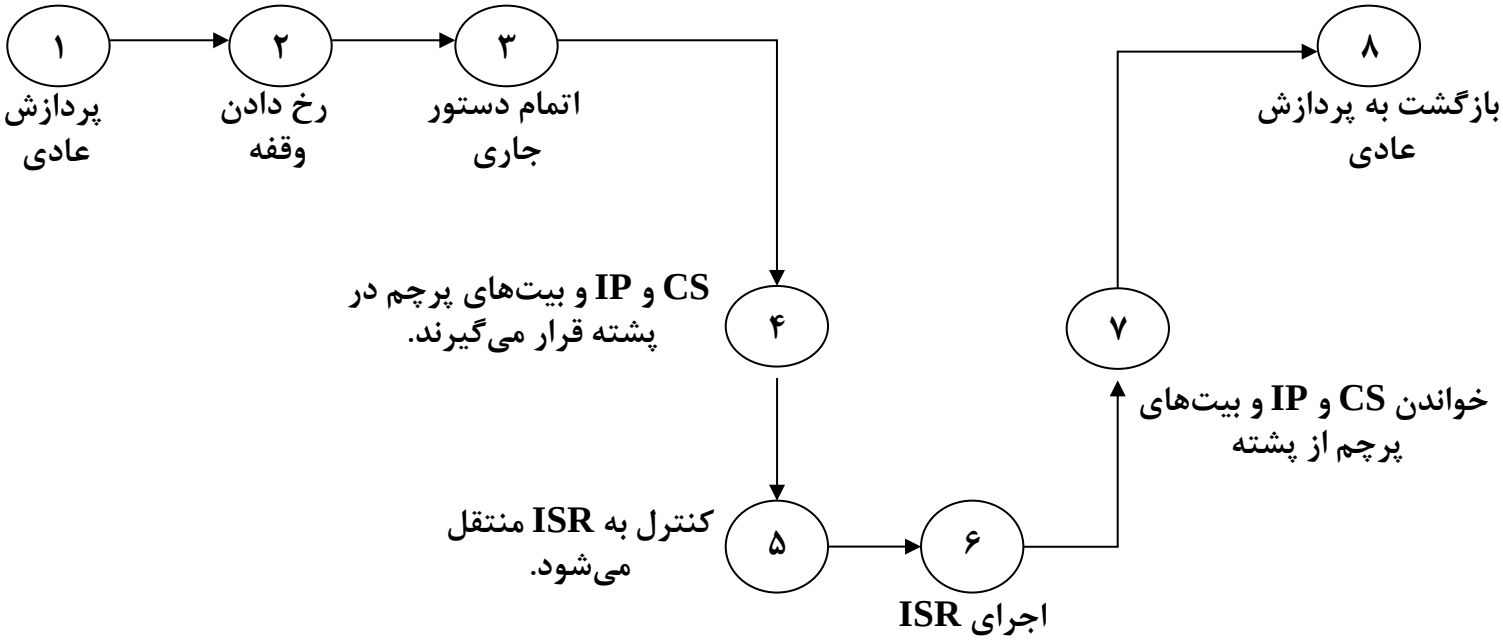
- مشکل اساسی در برقراری ارتباط بین ریزپردازنده و دستگاه جانبی در این است که پروسسور نمی‌داند کی دستگاه آماده است. در واقع دستگاه به صورت همگام با سیستم کار می‌کند.
- روش سرکشی برای حل این مشکل پیشنهاد چک کردن مداوم بیت وضعیت را داد که مشکلات آن را بررسی کردیم.
- بهتر است دستگاه جانبی به CPU اطلاع دهد که آماده است و سپس CPU شروع به سرویس‌دهی آن دستگاه کند. این اساس وقفه در سیستم‌های مبتنی بر ریزپردازنده است.
- در انتهای اجرای هر دستوری CPU خط وقفه را چک می‌کند و اگر فعال بود کنترل برنامه به مکان خاصی از حافظه که روتین سرویس وقفه (ISR: Interrupt Service Routine) نامیده می‌شود منتقل می‌شود.

I/O وقفه‌گرا (ادامه)

- شکل ۱۶ پاسخ CPU به یک وقفه را نشان می‌دهد.
- در مرحله‌ی ۱ فرض شده CPU پردازش عادی خود را انجام می‌دهد.
- در مرحله ۲ سیگنال READY دستگاه جانبی وقفه تولید کرده است. بعد از اتمام دستورالعمل جاری در مرحله‌ی ۳، محتوای رجیسترهای CP، IP و بیت‌های پرچم در پشته قرار می‌گیرند (مرحله‌ی ۴)، و مرحله‌ی ۵ انتقال کنترل برنامه به ISR است.
- بعد از اجرای روتین وقفه در مرحله‌ی ۶، محتوای پشته که رجیسترهای IP،CS و بیت‌های پرچم است از پشته بازیابی می‌شود و مرحله‌ی ۷ به اتمام می‌رسد. در مرحله‌ی ۸ پردازش عادی CPU ادامه می‌یابد.
- اگر فرض کنیم 100µs زمان برای پاسخ‌دهی به وقفه و فراهم کردن داده برای دستگاه جانبی لازم باشد و نیز نرخ دریافت داده‌ی چاپگر ۱۰۰ کاراکتر در ثانیه باشد (1/100=10000 µs) باشد آنگاه 9900 µs برای CPU باقی می‌ماند تا به پردازش عادی خود بپردازد. بدین ترتیب CPU می‌تواند چندین عمل را همزمان انجام دهد.

۸۰۸۶ دارای دو پایه وقفه‌ی INTR و NMI است.

I/O وقفه‌گرا (ادامه)



شکل ۱۶ فرآیند انجام شونده بعد از رخ دادن وقفه

انواع وقفه

- ۸۰۸۶ دارای هفت نوع وقفه‌ی متفاوت می‌باشد که در جدول ۴ آورده شده‌اند. NMI و INTR وقفه‌های خارجی هستند که از طرف سخت‌افزار اعمال می‌شوند. INT n، INTO و دستور INT3 وقفه‌های نرم‌افزاری هستند که در صورت نیاز در یک برنامه قرار داده می‌شوند.

- **وقفه‌های تقسیم بر صفر و تک گامی** را خود CPU اعمال می‌کند.

- اولی در صورتیکه حاصل عملیات تقسیم بیش از ظرفیت ثبات مقصد باشد به وجود می‌آید و دومی بعد از اجرای هر دستور در صورتیکه TF=1 باشد.

- در همه‌ی موارد دستورالعمل جاری قبل از پاسخگویی به وقفه **به اتمام می‌رسد**.

- **وقفه‌های داخلی به جز وقفه‌ی تک گامی** نسبت به وقفه‌های خارجی اولویت دارند.

- اگر **همزمان** وقفه‌هایی بر NMI و INTR اتفاق افتد، **ابتدا وقفه‌ی** NMI پاسخ داده می‌شود.

انواع وقفه‌های ۸۰۸۶

جدول ۴

تأخیر وقفه	آدرس جدول بردار	سیگنال تصدیق وقفه	اولویت	چگونگی تحریک	قابل پوشش؟	نحوه‌ی راه‌اندازی	نام وقفه
دستور جاری + ۵۱ (پالس‌ساعت)	00008H-0000BH	ندارد	2	لبه‌ی پالس، حداقل ۲ پالس پالس‌ساعت	خیر	سخت‌افزار خارجی	NMI
دستور جاری + ۶۱ (پالس‌ساعت)	$n^* \times 4$	$\overline{\text{INTA}}$	3	سطح بالا تا زمانی‌که تأیید شود	بلی از طریق IF	سخت‌افزار خارجی	INTR
۵۱ (پالس‌ساعت)	$n \times 4$	ندارد	1	ندارد	خیر	داخلی، نرم-افزاری	int n
۵۲ (پالس‌ساعت)	0000CH-0000FH	ندارد	1	ندارد	خیر	داخلی، نرم-افزاری	int 3
۵۳ (پالس‌ساعت)	00010H-00013H	ندارد	1	ندارد	بلی از طریق OF	داخلی، نرم-افزاری	into
۵۱ (پالس‌ساعت)	00000H-00003H	ندارد	1	ندارد	خیر	داخلی CPU	تقسیم بر 0
۵۱ (پالس‌ساعت)	00004H-00007H	ندارد	4	ندارد	بلی از طریق TF	داخلی CPU	تک‌گامی

همه‌ی انواع وقفه محتوای ثبات‌های CP و IP و نیز بیت‌های پرچم را در پشته قرار می‌دهند، به علاوه محتوای IF و TF نیز پاک می‌شود.

*** n یک عدد ۸ بیتی است که حین پالس دوم INTA از طریق باس داده خوانده شده و توسط وسیله وقفه دهنده تامین می‌شود.**

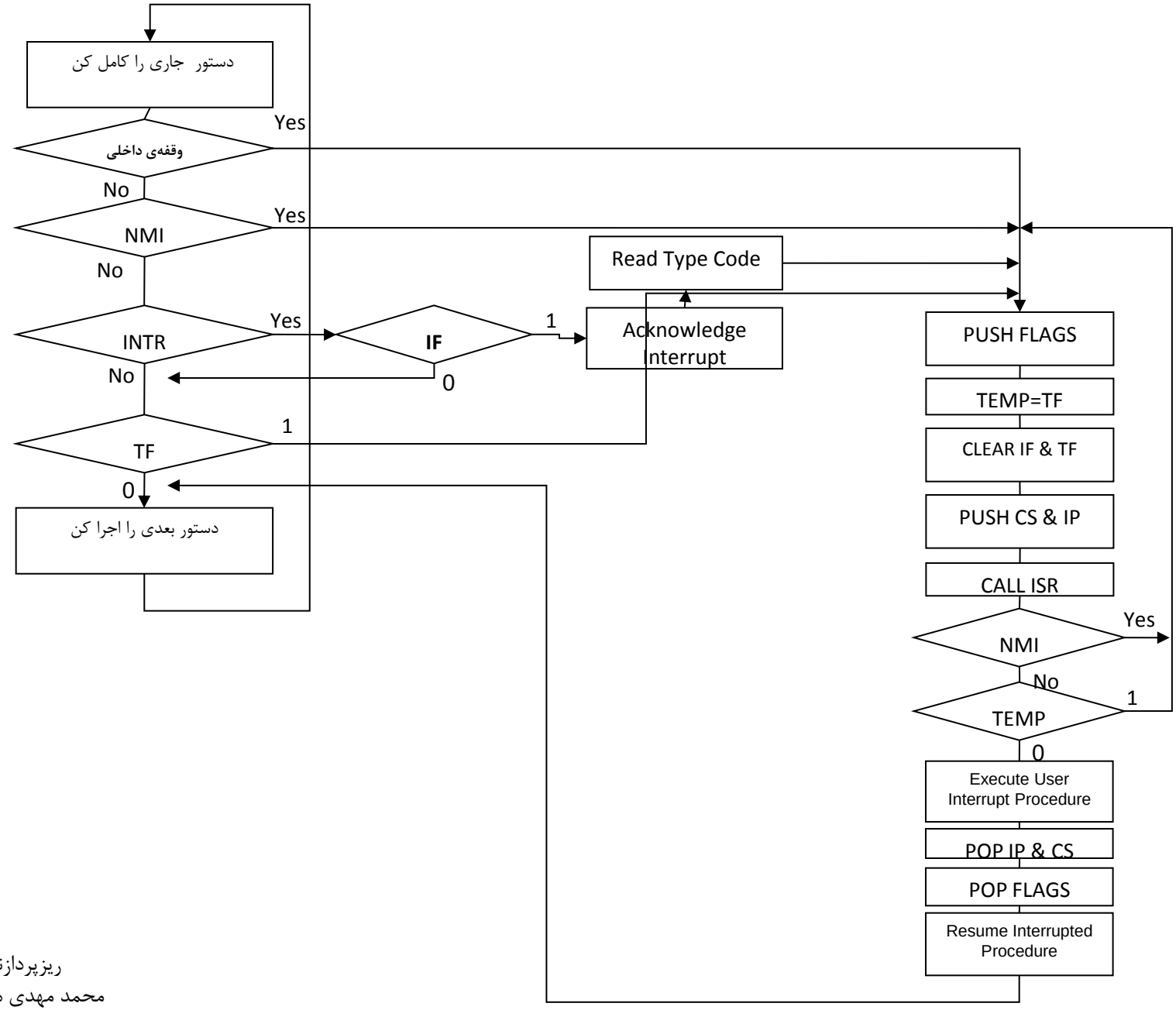
چنانچه دستوری بنام INT اجرا شود وقفه نوع ۳ یعنی INT 3 اجرا می شود.

[1] Interrupt Acknolwledge

انواع وقفه‌های ۸۰۸۶

- چنانچه تقسیم بر صفر رخ دهد وقفه نوع 0 یعنی وقف INT 0 رخ می‌دهد.
- پردازش تک گام وقفه نوع 1 است.
- چنانچه پایه NMI فعال شود وقفه نوع 2 یعنی INT 2 اجراء می‌شود.
- چنانچه دستوری بنام INT اجرا شود وقفه نوع 3 یعنی INT 3 اجرا می‌شود.
- INTO وقفه نوع 4 است یعنی INT4 اجرا می‌شود.

دنباله‌ی پردازش وقفه



انواع وقفه (ادامه)

- وقتی یک وقفه پاسخ داده می‌شود، محتوای رجیسترهای CS، IP و بیت‌های پرچم در پشته قرار می‌گیرند و لذا **شش بایت** در پشته قرار می‌گیرند.
- اگر پرچم TF=1 باشد، بعد از این فرآیند همچنان در وضعیت تک گامی می‌ماند.
- به هرحال قبل از اجرای ISR، محتوای TF و IF پاک می‌شوند و لذا INTR غیرفعال می‌شوند.
- اگر بخواهیم این موارد درون روتین وقفه فعال باشند می‌توانیم در بین دستورات روتین ISR پرچم‌های مربوطه را فعال کنیم.
- وقفه‌های همزمان بدین معنی است که دو یا چند وقفه با هم صورت گیرند که در این صورت با توجه به جدول اولویت وقفه‌ها، پاسخ‌گویی به اولویت بالاتر انجام می‌شود.
- اگر درون روتین سرویس‌دهی به یک وقفه باشیم و پرچم‌های وقفه را فعال کرده باشیم، حتی اگر یک وقفه با اولویت کمتری نسبت به وقفه‌ای که در حال حاضر به آن پاسخ می‌دهیم اتفاق افتد میکرو به آن پاسخ می‌دهد.

انواع وقفه (ادامه)

- NMI یک وقفه‌ی غیر قابل پوشش است بدین معنی که نمی‌توان مانع از آن شد. از طرف دیگر وقفه‌ی INTR از طریق پرچم IF قابل پوشیده شدن است. (TF-> IF در جزوه اصلاح شود)

- تنها زمانی که پرچم IF = 1 باشد وقفه‌های اعمال شده بر پین INTR پذیرفته می‌شود.

- اگرچه **وقفه‌های داخلی بر خارجی اولویت** دارند، به محض اینکه روتین سرویس وقفه‌ی داخلی شروع شد، در خواست وقفه‌ی NMI پذیرفته می‌شود. این مساله برای وقفه‌ی INTR صادق نیست چون به محض ورود به روتین ISR پرچم IF به طور خودکار 0 می‌شود.

- از آنجا که وقفه‌ی NMI را نمی‌توان در برنامه غیرفعال کرد، هنگام استفاده از این وقفه باید مراقب بود به خصوص در برنامه‌هایی که نباید در حین اجرا دچار وقفه شوند مانند خواندن یا نوشتن در دیسک درایوها.

- به همین دلیل کاربرد این وقفه موارد خاصی مثل رخدادن خطایی در حافظه یا ایراد قریب الوقوع در تغذیه سیستم است.

انواع وقفه(ادامه)

مثال: فرض کنید TF=1 و IF=1 و به طور همزمان وقفه‌های NMI و INTR اتفاق می‌افتد. کدام وقفه پذیرفته می‌شود؟ آیا روتین وقفه در مد تک گامی اجرا می‌شود؟

حل: شکل ۱۸ دستورالعمل‌های انجام شده را نشان می‌دهد. بعد از اتمام دستورالعمل جاری سه وقفه در حالت معوق قرار دارند: NMI، INTR و تک گامی. بالاترین اولویت به NMI اختصاص دارد و لذا ابتدا این وقفه تشخیص داده می‌شود. ولی به هرحال بعد از قرار دادن CS، IP و بیت‌های پرچم در پشته وقفه‌ی تک گامی تشخیص داده می‌شود (چون TEMP=1 است).

در واقع زیربرنامه تک‌گامی شدن دستورالعمل جاری قبل از روتین وقفه‌ی NMI فراخوانی می‌شود و می‌توان گفت **وقفه‌ی تک گامی بالاترین اولویت را دارد.**

بعد از اجرای روتین وقفه‌ی مربوطه بیت‌های پرچم به وضعیت‌شان قبل از وقفه‌ی تک گامی برمی‌گردند (TF=0 و IF=0)و وقفه‌ی NMI سرویس‌دهی می‌شود. چون TF,IF=0 لذا هیچ کدام از وقفه‌های INTR و تک گامی در این روتین تشخیص داده نمی‌شوند

انواع وقفه (ادامه)

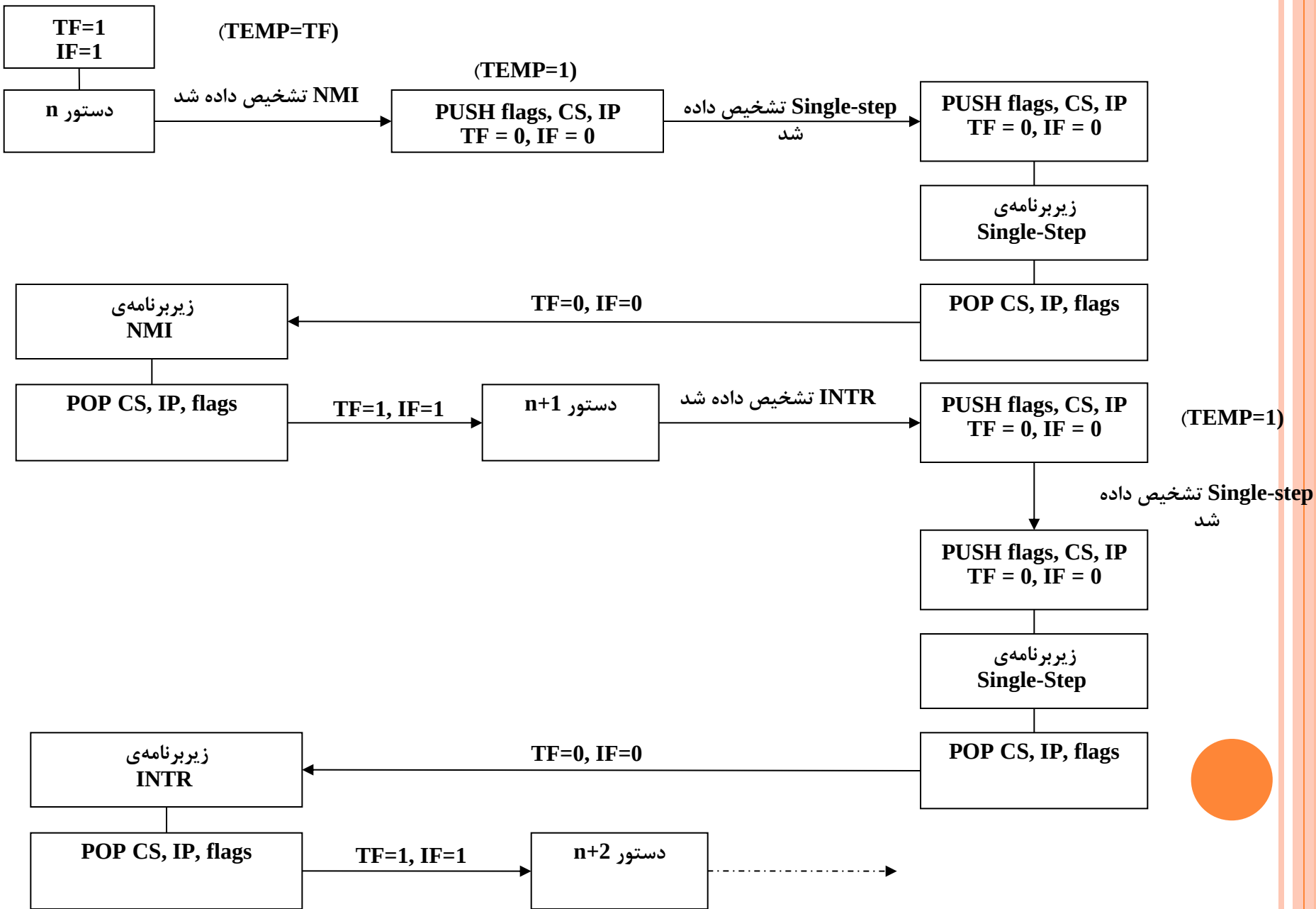
- بعد از اجرای دستور IRET بیت‌های پرچم به مقادیر اصلی خود بازمی‌گردند ولی این کار قبل از اجرای دستورالعمل بعدی انجام نمی‌شود. بنابراین یکی دیگر از دستورالعمل‌های برنامه‌ی اصلی اجرا می‌شوند و سپس وقفه‌ی INTR تشخیص داده می‌شود.

- بعد از تشخیص INTR بیت‌های پرچم، CS و IP درون پشته قرار می‌گیرند و IF,TF=0 می‌شوند ولی مجددا چون TEMP=1 است ابتدا روتین تک‌گامی شدن برای دستورالعمل دوم برنامه‌ی اصلی نیز اجرا می‌شود.

- به محض کامل شدن، زیربرنامه‌ی INTR اجرا می‌شود و مجددا چون TF=0 است این روتین در مد تک گامی نخواهد بود.

- دستور STI نیز همانند دستور IRET وقفه‌ی INTR را فعال نخواهد کرد تا زمانیکه دستورالعمل بعدی اجرا شود.

فرآیند انجام شونده با رخداد همزمان وقفه‌های NMI، INTR و با فعال بودن پرچم وقفه تک گامی



انواع وقفه (ادامه)

- آدرس هر روتین سرویس وقفه در چهار مکان متوالی حافظه در جدول بردار وقفه که در آدرس 00000H شروع می‌شود قرار دارد.

- همه انواع وقفه که در جدول ۴ لیست شده‌اند به جز INTR عددی هشت بیتی به عنوان عدد نوع وقفه نیز درون دستورالعمل فراهم می‌کنند یا اینکه یک عدد نوع از پیش تعریف شده دارند که به یکی از ۲۵۶ آدرس روتین وقفه در این جدول بردار وقفه اشاره می‌کند.

- ۸۰۸۶ برای پیدا کردن بردار ویژه‌ای که برای وقفه‌ی مربوطه باید به کار گیرد، عدد نوع وقفه را در ۴ ضرب می‌کند. عدد دو بیتی حاصل به یکی از ۲۵۶ بردار چهار بایتی اشاره می‌کند.

انواع وقفه (ادامه)

مثال: عدد نوع وقفه‌ی خاصی برابر n=41H است. اگر آدرس روتین وقفه‌ی مربوطه 09E3:0010H باشد، مکانی از جدول بردار وقفه که این آدرس را در خود نگه می‌دارد را بیابید.

حل: آدرس بردار وقفه از ضرب عدد نوع در 4 بدست می‌آید که برابر است با 104H یا 00104H. IP در آدرس کم ارزش و CS در آدرس پرارزش قرار می‌گیرد.

00107H: 09H

00106H: E3H

00105H: 00H

00104H: 10H

- وقفه‌ی NMI به طور پیش فرض از نوع 2 تعریف شده است و لذا آدرس بردار مربوطه 00008:0000BH خواهد بود. به هرحال INTR باید عدد نوع خود را طی سیکل باس خاصی که اعلام وقفه نامیده می‌شود بر خطوط باس داده‌ی AD0-AD7 قرار دهد.

زمانبندی وقفه‌ی خارجی

- ۸۰۸۶ ورودی‌های INTR و NMI را در آخرین کلاک از آخرین سیکل باس هر دستورالعمل نمونه‌برداری می‌کند. **NMI حساس به لبه‌ی بالا رونده است** و به طورداخلی سنکرون شده است.

- برای تضمین شناخته شدن وقفه از طرف CPU **حداقل به مدت دو برابر زمان کلاک** باید در وضعیت HIGH قرار داشته باشد.

- ورودی INTR حساس به سطح است و باید در سطح HIGH قرار داشته باشد تا زمانی‌که تشخیص آن از طرف CPU اعلام شود.

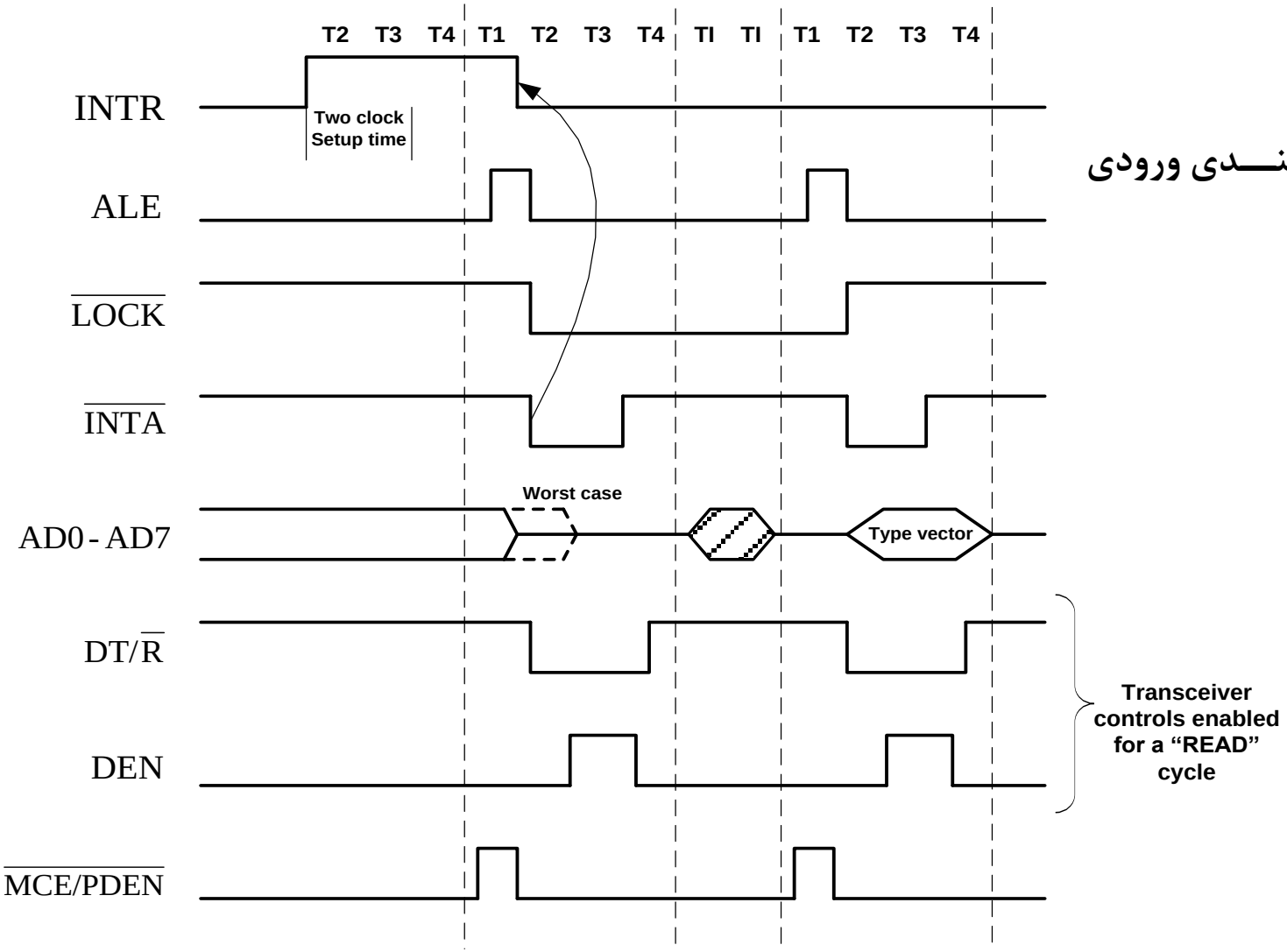
- $\overline{INTA}$سیگنال خروجی CPU برای اعلام تشخیص وقفه است. وقتی $\overline{INTA}=0$است ۸۰۸۶ وقفه را تشخیص داده و پذیرفته است و آن هم زمانی رخ می‌دهد که IF=1 باشد.

- سیگنال $\overline{INTA}$ تنها برای وقفه‌ی INTR استفاده می‌شود و برای دیگر وقفه‌های داخلی و نیز NMI اعلامی صورت نمی‌گیرد.

زمانبندی وقفه‌ی خارجی (ادامه)

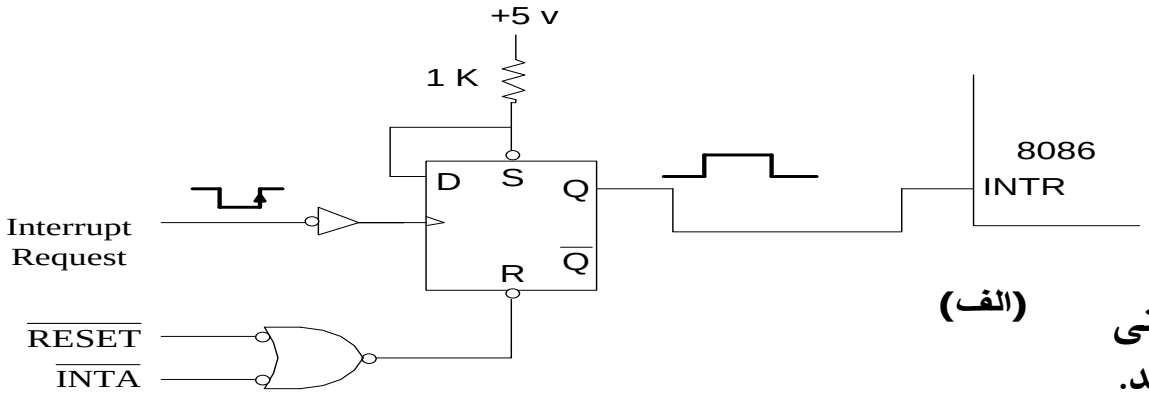
- شکل ۱۹ زمانبندی سیکل باس اعلام وقفه را نشان می‌دهد.
- INTR باید قبل از کلاک T4 از دستورالعمل وقفه یافته، زمان راه‌اندازی به مدت دو کلاک را برآورده کند.
- اگر این شرایط برقرار نباشد، وقفه پذیرفته نمی‌شود تا پایان دستورالعمل بعدی.
- نکته اینکه این زمان انتظار ممکن است بیش از ۱۰۰ کلاک به طول انجامد که این مساله در مورد دستورات ضرب و تقسیم رخ می‌دهد.
- بعد از پذیرفته شدن وقفه دو سیکل اعلام تصدیق وقفه اجرا می‌شود که با دو سیکل باس بیکار از هم جدا شده‌اند.
- **اولین پالس** ، نوع وقفه را درخواست می‌کند و به سخت‌افزار خارجی اعلام می‌کند که برای قرار دادن نوع وقفه بر روی خطوط باس داده آماده شود.
- **در حین دومین پالس**، CPU محتوای خطوط AD0-AD7 را می‌خواند و این داده را به عنوان یکی از ۲۵۶ عدد ممکن برای بیان نوع وقفه تفسیر می‌کند.

زمانبندی وقفه‌ی خارجی (ادامه)



شـــکل ۱۹- زمانبنـــدی ورودی INTR

زمانبندی وقفه‌ی خارجی(ادامه)

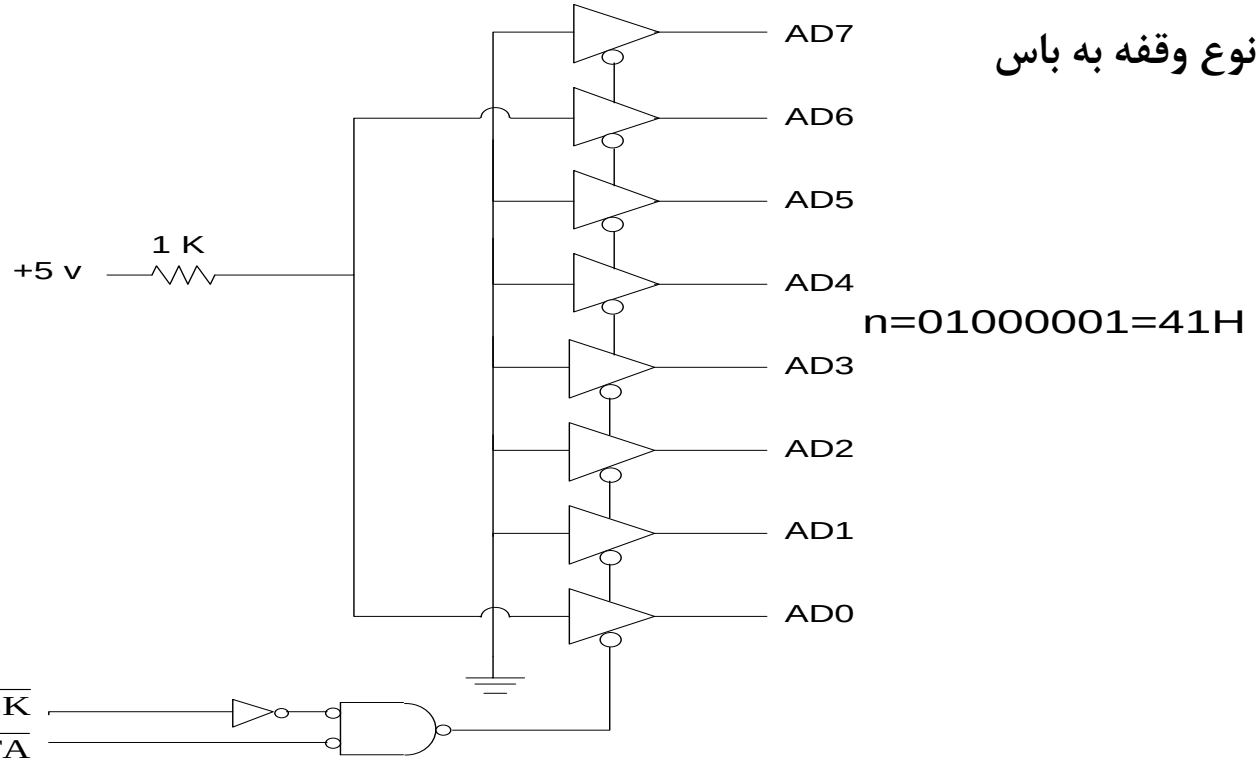


(الف)

شکل ۲۰

الف: خاتمه دادن به INTR وقتی درخواست وقفه پاسخ داده شد.

ب: تحویل عدد نوع وقفه به باس داده

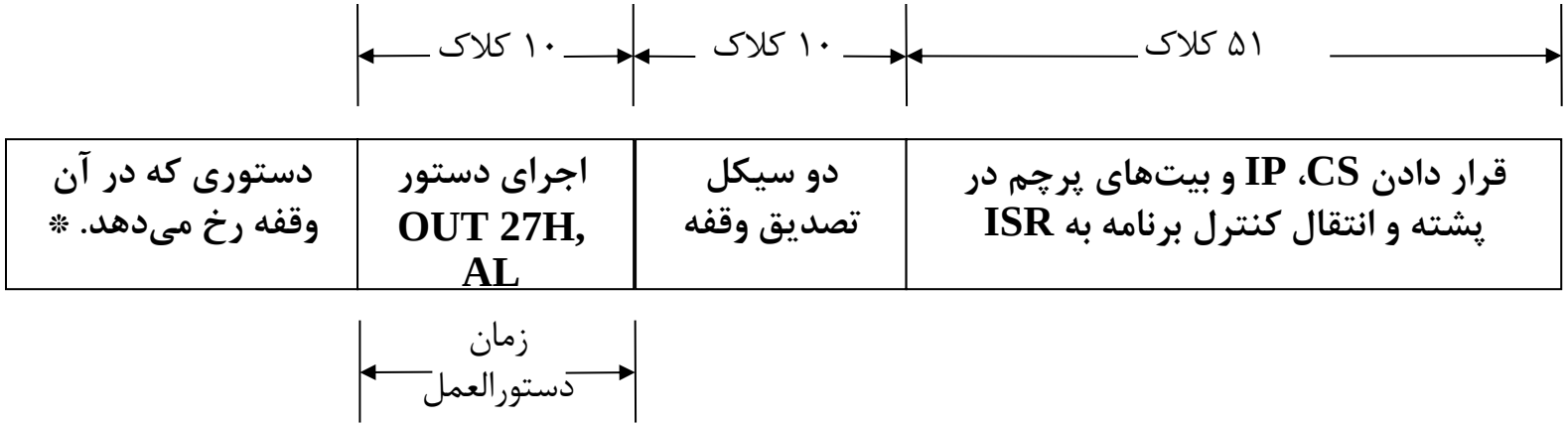


(ب)

زمان پاسخگویی

- حلقه‌ی اصلی روش سرکشی به ۳۰ کلاک زمان برای اجرا نیاز داشت. لذا پاسخگویی به هر دستگاه جانبی 6µs به طول می‌انجامد و البته زمانی که چندین دستگاه باید سرکشی شوند، این زمان بیشتر خواهد شد.
- زمان پاسخگویی وقفه یا تاخیر وقفه شامل زمان لازم برای انجام کارهای زیر است:
  - خاتمه دادن اجرای دستور فعلی
  - اجرای دو سیکل باس تصدیق وقفه
  - قرار دادن محتوای IP، CS و بیت‌های پرچم در پشته
  - محاسبه‌ی آدرس جدول بردار وقفه و انتقال کنترل برنامه به روتین ISR
- شکل ۲۱ مثالی را نشان می‌دهد که در آن وقفه درست قبل از آنکه دستور OUT 27H, AL اجرا شود رخ می‌دهد.
- با فرض اینکه شرط زمان راه اندازی برقرار نبوده باشد، CPU باید ۱۰ کلاک برای اجرای دستور جاری منتظر بماند، ۱۰ کلاک دیگر برای سیکل‌های تصدیق وقفه و ۵۱ سیکل دیگر هم برای قرار دادن CS، IP و بیت‌های پرچم در پشته صرف می‌شود.
- این زمان در یک سیستم با کلاک 5MHz برابر با 14.2 µs خواهد بود.

زمان پاسخگویی (ادامه)



*: وقفه در این دستور زمان راه‌اندازی را از دست داده است.

زمان پاسخگویی به وقفه (تاخیر) = $1/f_{clock}$ *(۶۱+ $T_{instruction}$)

زمان پاسخگویی (ادامه)

- در بدترین حالت این زمان می‌تواند بسیار طولانی‌تر شود. مثلا اجرای دستور ROR [BX+DI+7], CL وقتی CL=FFH باشد به مدت زمان $T=(20+EA+4*CL)*{}^{1}/_{f}$ برای اجرا نیاز دارد.

- مد آدرس‌دهی پایه به همراه شاخص به ۱۲ کلاک برای **محاسبه‌ی آدرس موثر (EA)** نیاز دارد. و به ازای CL=255 **جمعا ۱۰۵۲ کلاک مورد نیاز** است و **زمان پاسخ گویی 210.4 µs** خواهد شد.

- **وقفه‌ی NMI سیکل‌های اعلام وقفه را انجام نمی‌دهد** و لذا تاخیر آن ۱۰ کلاک کمتر خواهد بود و لذا در مورد قبلی زمان پاسخگویی به 208.4µs می‌رسد.

- به نظر می‌رسد زمان پاسخگویی وقفه خیلی بیشتر از روش سرکشی شده است که با هدف اولیه‌ای که باعث شد به سراغ این روش آئیم مغایرت دارد!

- لازم است به این نکته توجه داشته باشیم که 210.4µs کمتر از ۳٪ زمان لازم برای کار با چاپگری است که با سرعت ۱۰۰ کاراکتر بر ثانیه داده دریافت می‌کند و اگر ۱٪ زمان هم برای ارسال داده به چاپگر زمان لازم باشد ۹۶٪ باقی زمان برای انجام دیگر کارهای CPU باقی می‌ماند.

زمان پاسخگویی (ادامه)

- به این نکته توجه داشته باشید که **بعضی دستورات وقفه پذیر نیستند**. مثل دستورات:

MOV segment register یا **POP segment register**

- وقفه‌ای که در حین اجرای این دستورات اتفاق افتد پذیرفته نمی‌شود تا دستورالعمل بعد از دستور POP یا MOV. این حفاظت برای آن است که برنامه با یک سگمنت پشته‌ی جدید ولی اشاره‌گر پشته‌ی قدیمی پایان نیابد.

- پیشوندهای LOCK و تغییر سگمنت پیش‌فرض جزئی از دستوری که به دنبال آنها می‌آید در نظر گرفته می‌شوند و **لذا وقفه پذیر نیستند.**

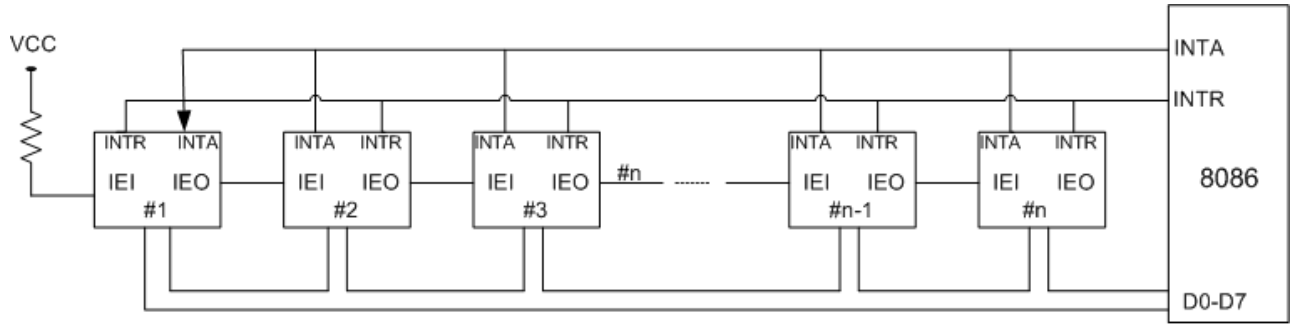
- پیشوند REP از این مساله استثناء می‌شود و لذا وقفه بین دستورات تکرارشونده‌ی کار با رشته‌ها پذیرفته می‌شود.

اولویت‌دهی وقفه‌ها

برای اولویت‌دهی به وقفه‌ها در صورت فعال شدن همزمان می‌توان روش‌های زیر را بکار برد:

1. **استفاده از encoder (مثل استفاده از encoder به شماره ۷۴۱۴۸)**
2. **روش Daisy Chain**

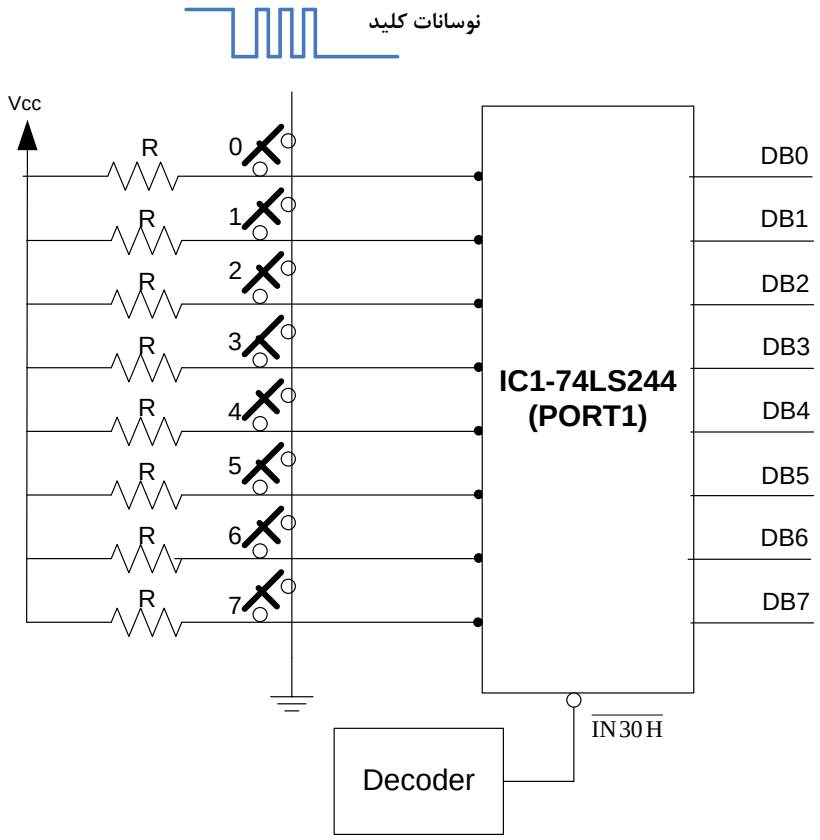
- **شکل زیر استفاده از روش daisy chain را نشان می‌دهد. وسایل جانبی با #n مشخص شده‌اند. هر یک از آنها می‌تواند از طریق خروجی INTR خود تقاضای وقفه کند.**
- **شرط اینکه یک وسیله بتواند تقاضای وقفه کند آن است که ورود IEI آن high باشد. اگر وسیله‌ای تقاضای وقفه کرد خروجی IEO خود را low می‌کند.**
- **وسیله با اولویت بیشتر که تقاضای وقفه نموده است با دریافت INTA عدد وقفه را از طریق باس داده در اختیار ریزپردازنده می‌گذارد.**



IEI: Interrupt Enable Input

IEO: Interrupt Enable Output

طراحی کیبورد سطری

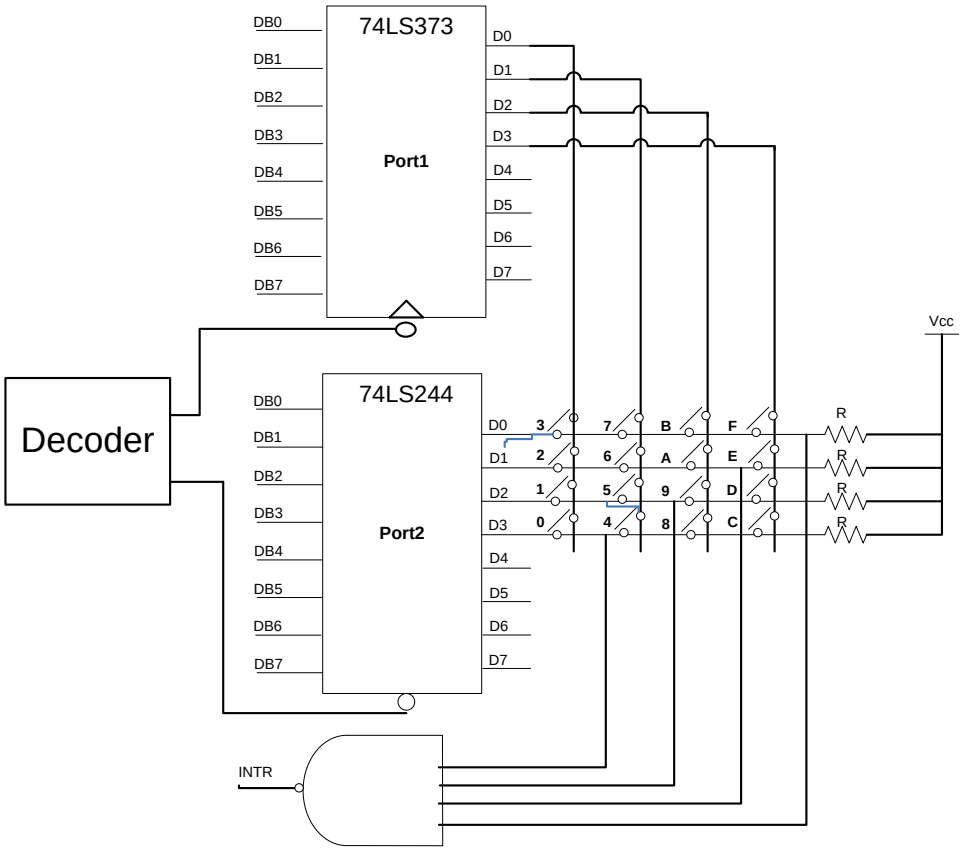


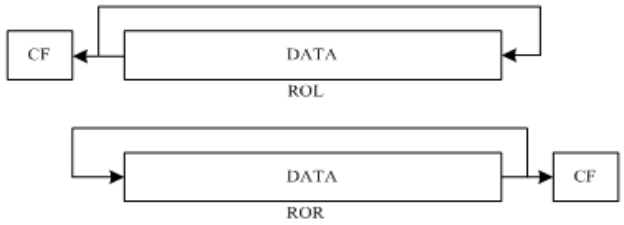

```
Loop1:      in al,      PORT1
            cmp         al, 0FFH
            jz          Loop1
            call        Delays20ms
            in          al, PORT1
            mov         cl, 07H
Loop2:      sal         al
            jnc         Label1
            dec         cl
            jnz         Loop2
Label1:     mov         al, cl
```

```
DelayShort:
Again1:     mov  cx, LoopCounter1
            nop
            loop  Again1
            ret
```

```
DelayLong:
Again2:     mov  bx, LoopCounter2
            call  DelayShort
            dec bx
            jnz Again2
            ret
```

طراحی کیبورد ماتریسی

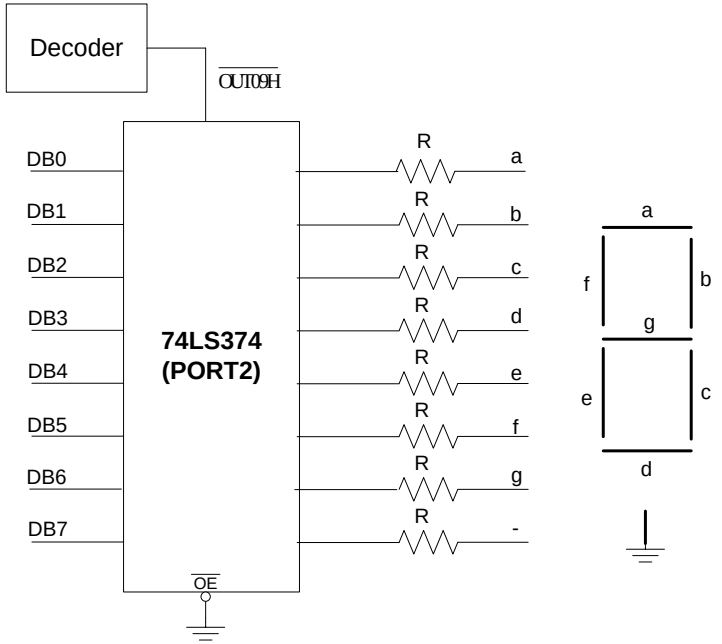





```
KeyStroke: mov al, 0x00
           out port1, al
loop0:     in al, port2
           and al, 0x0F
           cmp al, 0x0F
           jpz loop0
           call Delay20ms
           call keyfind
           ...

keyfind:   mov dl, 0x00
           mov dh, 0x00
           mov cl, 0b11111110

loop1:     mov al, cl
           out port1, al
           in al, port2
           mov bl, 0x04

loop2:     ror al, 1
           jnc label1
           dec bl
           jnz loop2
           add dh, 0x04
           rol cl, 1
           inc dl
           cmp dl, 0x04
           jnz loop1

label1:    mov al, bl
           add al, dh
           dec al
           ret
```

اتصال نمایش‌دهنده ۷ قطعه‌ای به ریزپردازنده

رقم	نام قطعه								معادل شانزدهي
	-	a	b	c	d	e	f	g	
0	0	1	1	1	1	1	1	0	7EH
1	0	0	1	1	0	0	0	0	30H
2	0	1	1	0	1	1	0	1	6DH
3	0	1	1	1	1	0	0	1	79H
4	0	0	1	1	0	0	1	1	33H
5	0	1	0	1	1	0	1	1	5BH
6	0	1	0	1	1	1	1	1	5FH
7	0	1	1	1	0	0	0	0	70H
8	0	1	1	1	1	1	1	1	7FH
9	0	1	1	1	1	0	1	1	7BH
A	0	1	1	1	0	1	1	1	77H
b	0	0	0	1	1	1	1	1	1FH
c	0	0	0	0	1	1	1	0	0EH
d	0	0	1	1	1	1	0	1	3DH
E	0	1	0	0	1	1	1	1	4FH
F	0	1	0	0	0	1	1	1	47H
خاموش	0	0	0	0	0	0	0	0	00H




```
MOV   AL, 7FH
OUT   09H, Al
```

نمایش رقم 8 بر روی نمایش‌دهنده

Table7Seg: db 7EH, 30H, 6DH, 79H, 33H, 5BH, 5FH, 70H, 7FH, 7BH, 77H, 1FH, 0EH, 3DH, 4FH, 47H, 00H