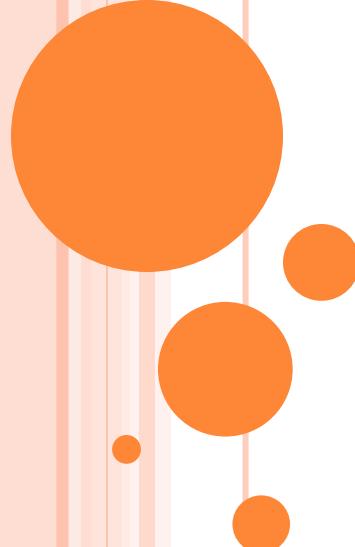


فصل ششم

روش های پایه‌ی 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 از آن خط منتقل می‌شوند. طبیعتاً روش سریال کندر 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 مجدداً به **۰** و **۱** تبدیل می‌کند. بدین ترتیب کامپیوتر می‌تواند با سیستمی در فاصله‌ی چند هزار کیلومتری ارتباط برقرار کند.

I/O موازی

- سخت افزار لازم برای پورت I/O موازی شبیه مدار واسط RAM و ROM است.
- زمانیکه CPU یک دستور خروجی را اجرا می کند (سیکل نوشتن I/O)، پورت باید داده موجود بر باس را ذخیره کند.
- به طور مشابه وقتی دستور ورودی اجرا می شود (سیکل خواندن I/O) پورت باید داده را بر خطوط باس داده وارد کند.
- پورت های I/O آدرس های ویژه خود را دارند.

I/O زمان بندی سیکل باس

- ۸۰۸۶ و ۸۰۸۸ فقط دو دستور برای ورود و خروج داده دارد.
 - ✓ "شماره‌ی پورت IN AX، " برای خواندن از پورت
 - ✓ "شماره‌ی پورت OUT AL،" 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/O موازی (ادامه)

باس کنترل

نوع	دستور العمل	باس آدرس	باس داده	مد حداقل	مد حداقل
مستقیم	IN AL (or AX), پورت	آدرس پورت = A0-A7 = A8-A19=0	D0-D7= بايت زوج D8-D15= بايت فرد D0-D15= کلمه زوج	M/I0=0 $\overline{RD}=0$	IORC
	OUT AL(or AX), پورت	آدرس پورت = A0-A7 = A8-A19=0	D0-D7= بايت زوج D8-D15= بايت فرد D0-D15= کلمه زوج	M/I0=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

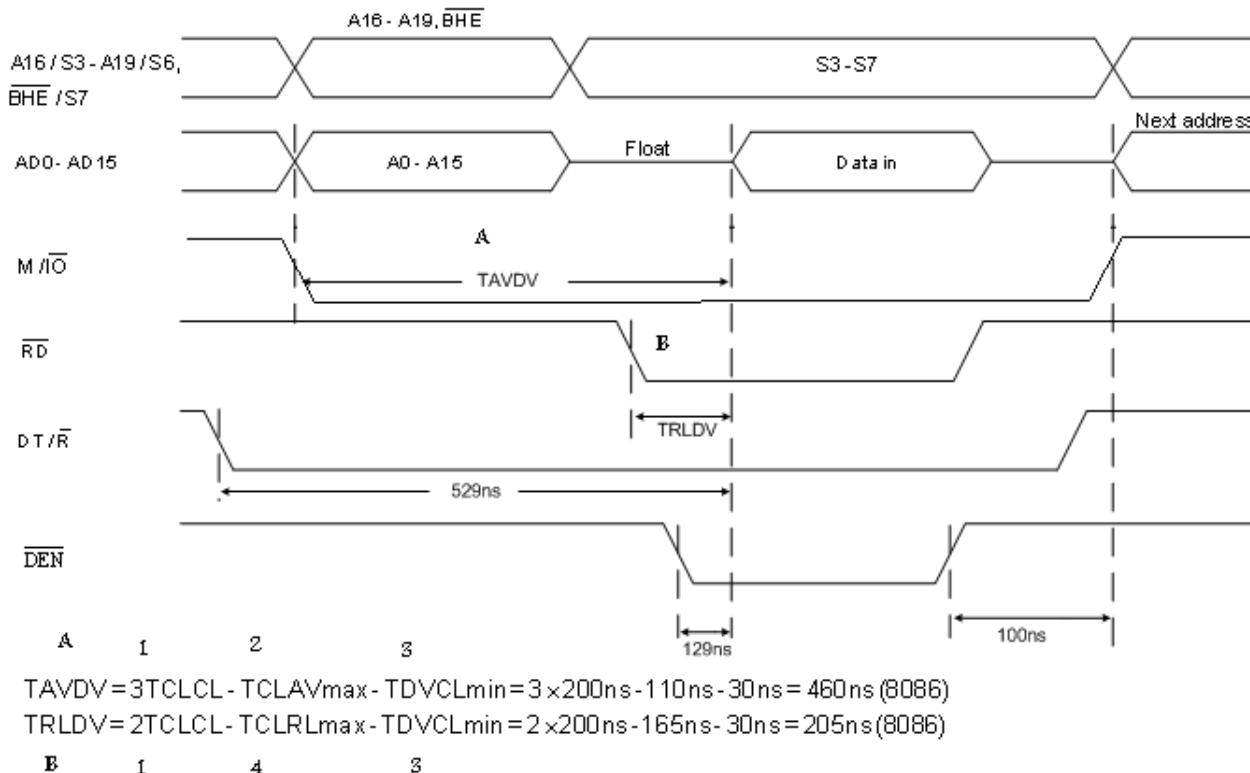
A0 و \overline{BHE} به صورت زیر کدگذاری می‌شوند:

\overline{BHE} A0	
0	0
0	1
1	0
1	1

- دسترسی کلمه‌ای
- دسترسی به بایت‌های فرد
- دسترسی به بایت‌های زوج
- بی تاثیر



زمانبندی I/O در سیکل خواندن (مد مینیمم)



TAVDV: Address Access Time

TCLCL: Clock cycle periode

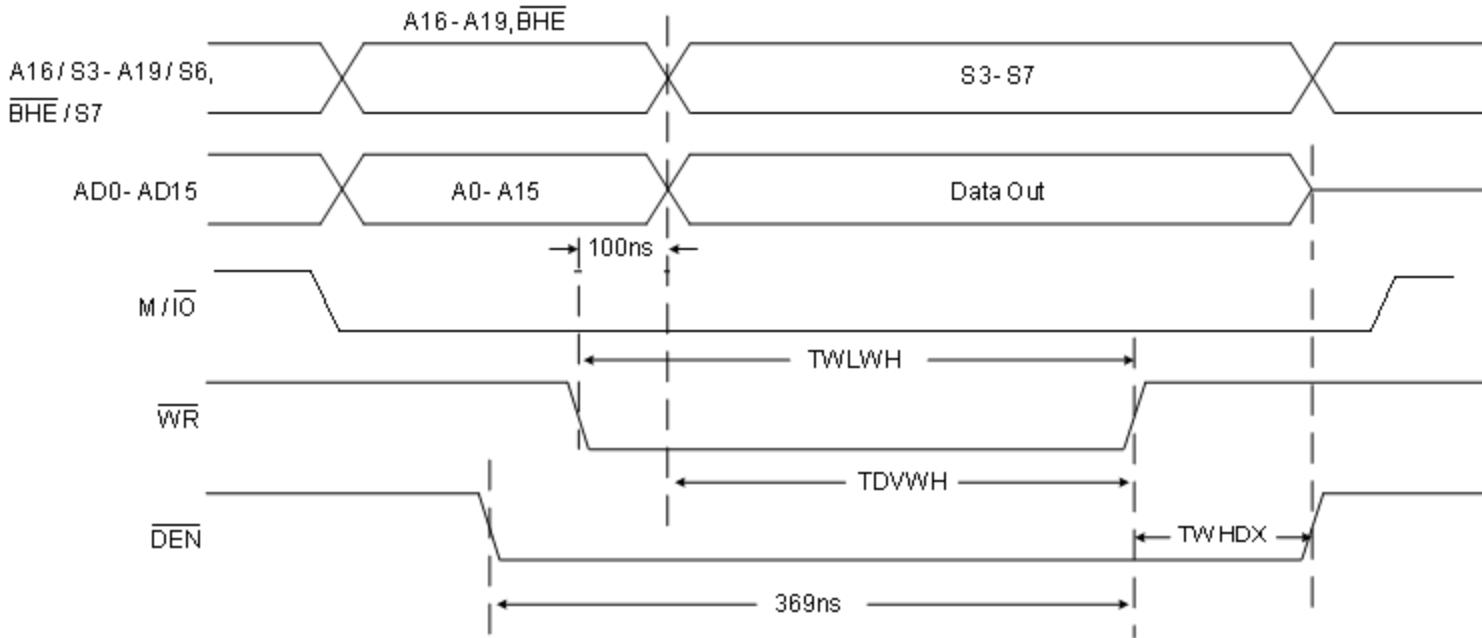
TCLAV: Address Valid Delay

TDVCL: Data Setup Time

TRLDV: Read Access Time

TCLRL: RD Active Delay

زمانبندی O/I در سیکل نوشتن (مد مینیمم)



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

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

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

$$TWHDX = TCLCH - 30\text{ns} = 118\text{ns} - 30\text{ns} = 88\text{ns} \text{ (8086)}$$

TCLCL: Clock cycle period

TCLDV: Data Valid Delay

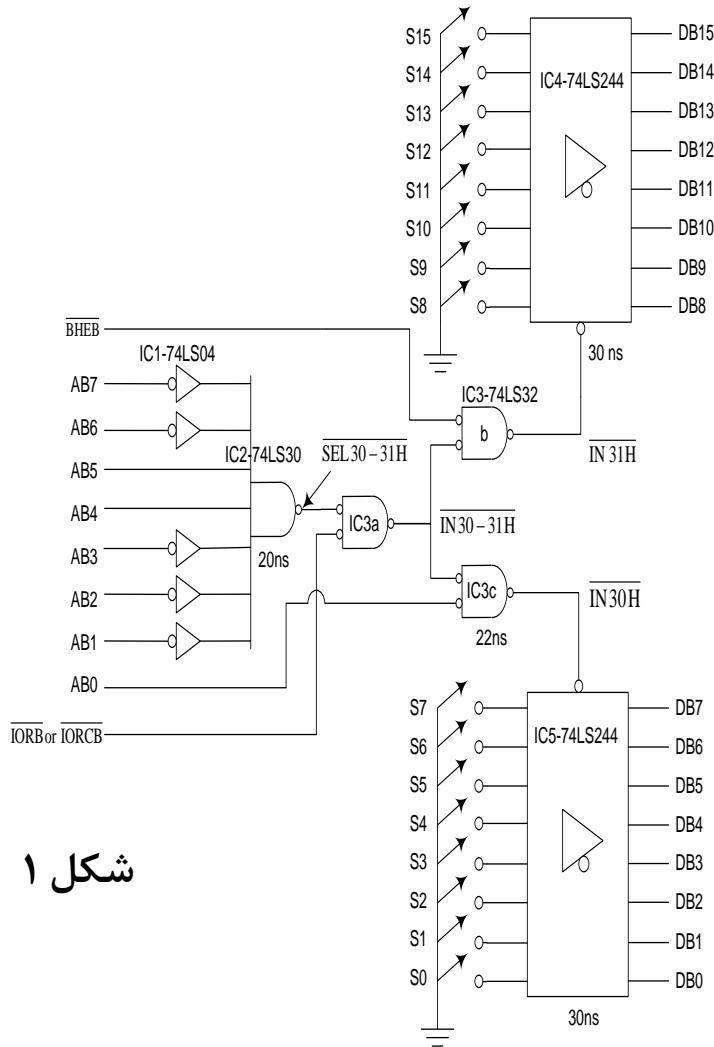
TCVCT: Control Active Dealy 1

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 می‌دهد:



شکل ۱

طراحی پورت ورودی موازی

- برای شبیه‌سازی داده‌ی ورودی، از ۱۶ سوئیچ استفاده می‌شود. IC1 و IC2 آدرس پورت (مستقیم) قرار گرفته بر AB7 تا AB0 را دیکود می‌کنند. خروجی IC2 سیگنال انتخاب پورت است.
- (DSP) تراشه‌ی IC3a سیگنالهای SEL 30 – 31H و IORB را با هم ترکیب می‌کند تا پالس انتخاب دستگاه را بسازد: IN 30 – 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.
- IN 30 – 31H با A0 و BHEB ترکیب می‌شوند تا پالس انتخاب دستگاه فرد یا زوج را به طور مجزا تولید کنند.
- این سیگنال‌ها نهایتاً به ورودی فعال‌ساز بافرهای سه حالته وصل می‌شوند و موجب قرار گرفتن داده بر خطوط بس 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 ۱ قرار می‌دهد.

باید در برنامه ۱ بودن بیت‌های خواسته شده را بررسی کنیم.

این کار با دستور **TEST AX, 280CH** انجام می‌شود که در صورت ۱ 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 2,3,11 or 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

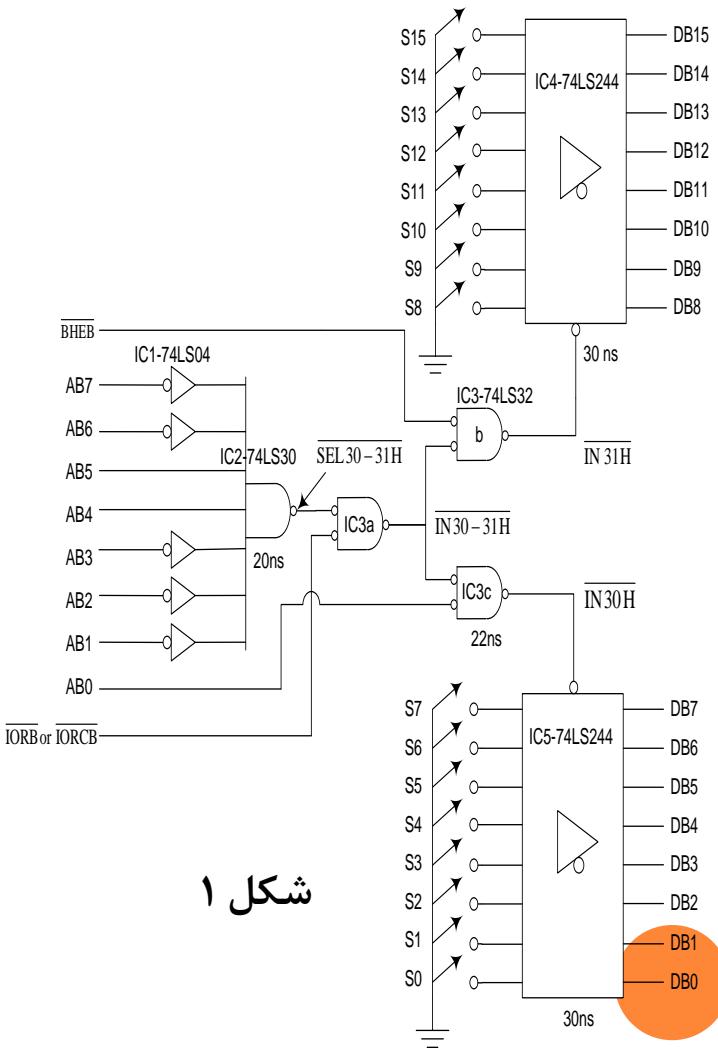
```

PUBLIC PROC1      ;comment
IPORT EQU 30H     ;input port = 0030H

0000      CODE    SEGMENT BYTE PUBLIC
0000          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
000A          END

```



شكل ١

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 **DB8-DB15** یا **DB0-DB7** 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, DX** (یا **IN AX, 31H**) 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.
- توان 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 TRLDV و TAVDV برای مدار حداقل ۸۰۸۶ با کلاک ۵MHz را برآورده می‌سازد. پیکربندی کاملاً 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 از بافرها و گیت‌هایی است که آدرس را دیکود می‌کنند.

Address Bus Buffer Delay

+tIC4

+30ns

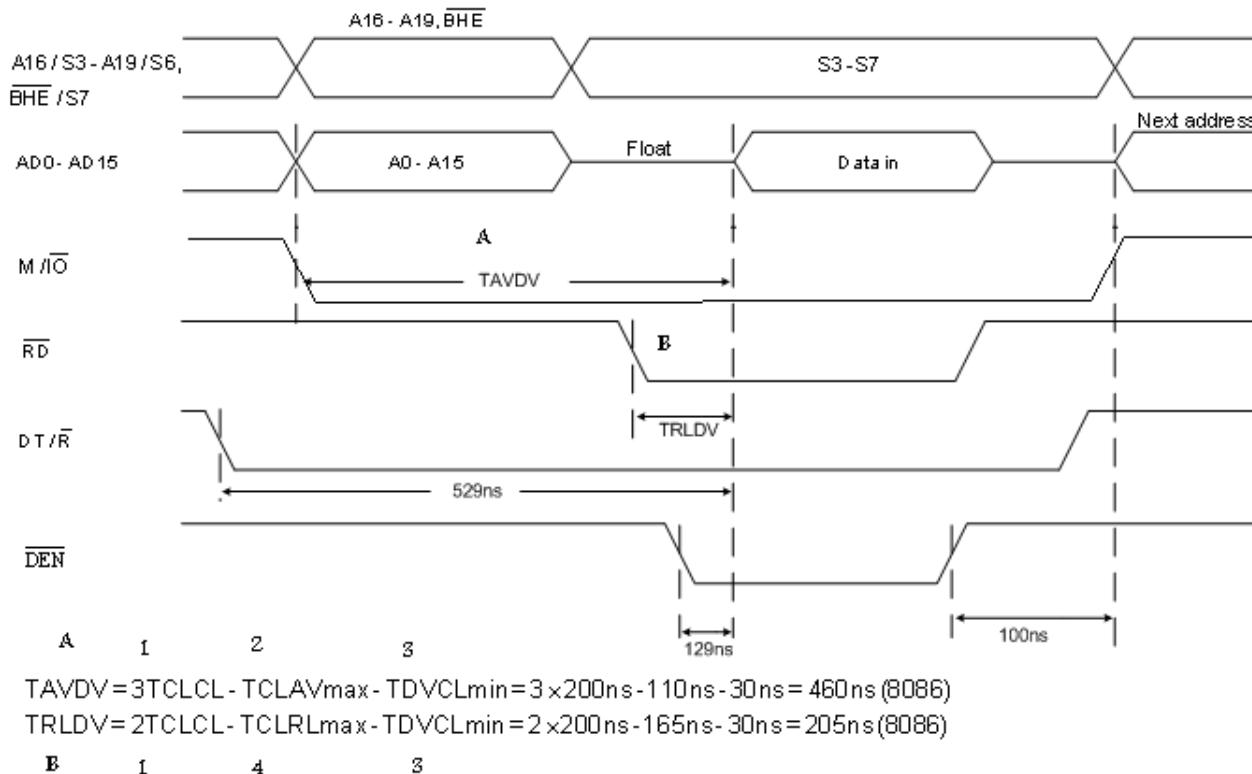
$$T = t_{buf} + t_{IC1} + t_{IC2} + t_{IC3a} + t_{IC3b-c} = (120 + 15 + 20 + 22 + 22) \text{ ns} = 229 \text{ ns}$$

چون مقدار بالا برای ریزپردازنده ۸۰۸۶ از ۴۶۰ns کمتر است پس قابل قبول است. چون TRLDV مقدار تاخیر ایجاد شده از قرار گرفتن 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:

$$T = t_{IC4} + t_{IC3a} + t_{IC3b-c} = (30 + 22 + 22) \text{ ns} = 74 \text{ ns}$$

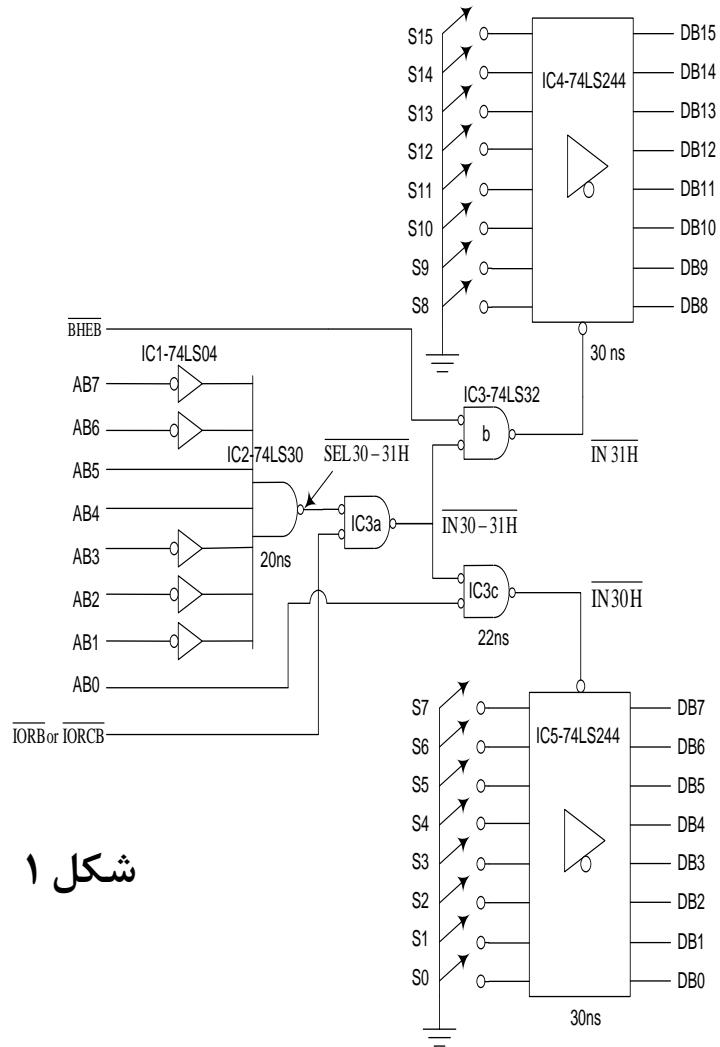
بر طبق مشخصات، حداقل مقدار بالا ۲۰۵ns است.

زمانبندی I/O در سیکل خواندن (مد مینیمم)



طراحی پورت ورودی موازی (ادامه)

مدار مثال ۱:



شکل ۱

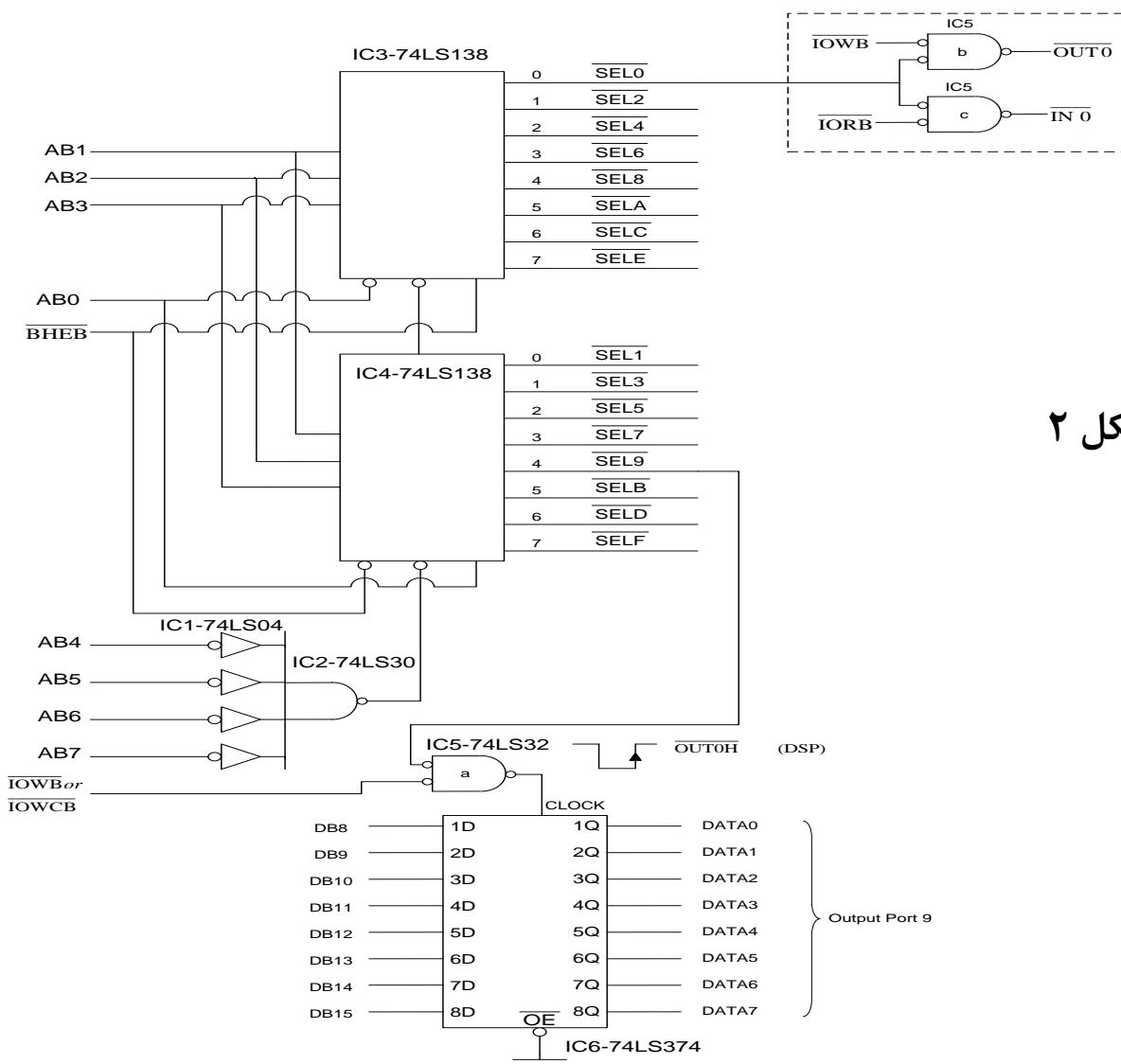


طراحی پورت خروجی موازی

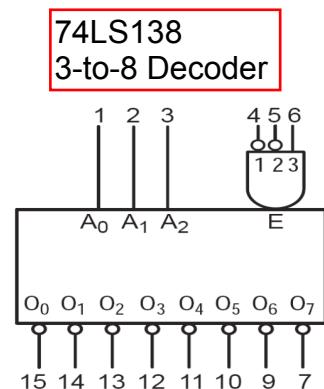
- سخت افزار مورد نیاز برای طراحی پورت خروجی، مشابه پورت ورودی است به جز اینکه سیگنال **Device Select Pulse (DSP)** به جای فرمان دادن به بافرهای سه حالت، به یک لچ فرمان می دهد.
- وجود لچ به این دلیل است که داده مدت زمان کوتاهی در خروجی **CPU** قرار می گیرد.
- شکل ۲، مداری برای فراهم کردن ۱۶ سیگنال انتخاب مجزا را فراهم می کند.
- IC4 و AB0 و $\overline{\text{BHEB}}$ به گونه ای سیم کشی شده اند که IC3 سیگنال انتخاب آدرس های زوج و آدرس های فرد را ایجاد می کند.



طراحی پورت خروجی موازی (ادامه)



شکل ۲



طراحی پورت خروجی موازی (ادامه)

- در این مورد خاص **IC5a** با $\overline{\text{SEL}\,9}$ و $\overline{\text{OUT}\,9}$ را تولیدمی‌کنند.
- لبه‌ی بالا رونده‌ی این سیگنال (و در واقع $\overline{\text{IOWB}}$) باعث می‌شود که **74LS374** داده‌ی موجود بر پورت **DB8-DB15** را لچ کند.
- دو دستور **AL OUT 9, AL** و **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 ۲، ۳، ۱۱ یا ۱۳ باز هستند. اگر شرایط برقرار بود، ۰۰ را در این پورت خروجی بنویسید. 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 ۰۰ 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 نوشتمندی 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 $t_{hold} = 0\text{ns}$ و $t_{setup} = 20\text{ns}$ است. بررسی کنید که این مشخصات در یک ماجول CPU کاملاً بافر شده با مد حداقل و کلاک ۵MHz ارضاء می‌شود.

حل: ۸۰۸۶ داده را TDVWH نانوثانیه قبل از لبه‌ی بالا رونده‌ی $\overline{\text{IOW}}$ خارج می‌کند. داده به خاطر بافرهای باس، ۶۰ نانوثانیه تاخیر می‌یابد. به هر حال لبه‌پشتی $\overline{\text{IOW}}$ نیز به همان مقدار تاخیر به اضافه‌ی IC5 تاخیر می‌یابد. در بدترین حالت لازم است که

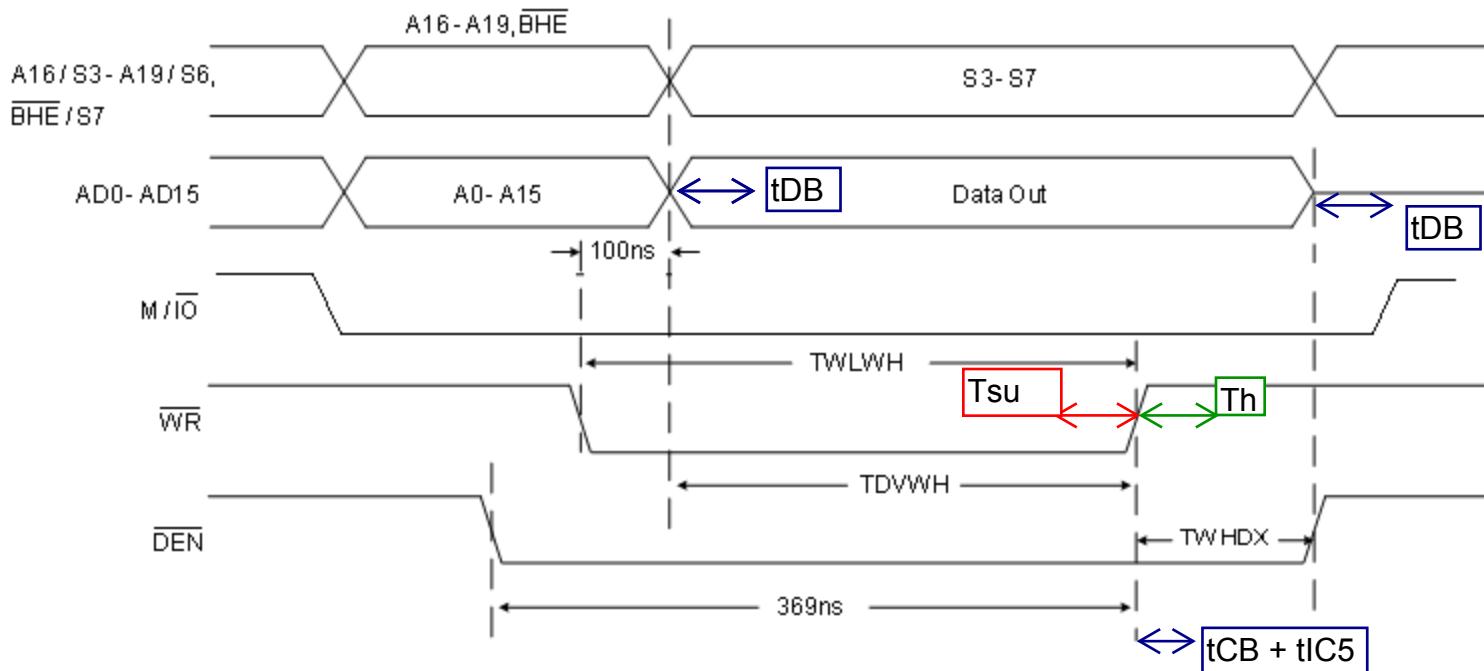
شرط نامساوی زیر برقرار باشد:

$$t_{su} \leq \text{TDVWH} - (t_{DBbufmax} - t_{CBbufmin} - t_{ICbufmin})$$

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



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

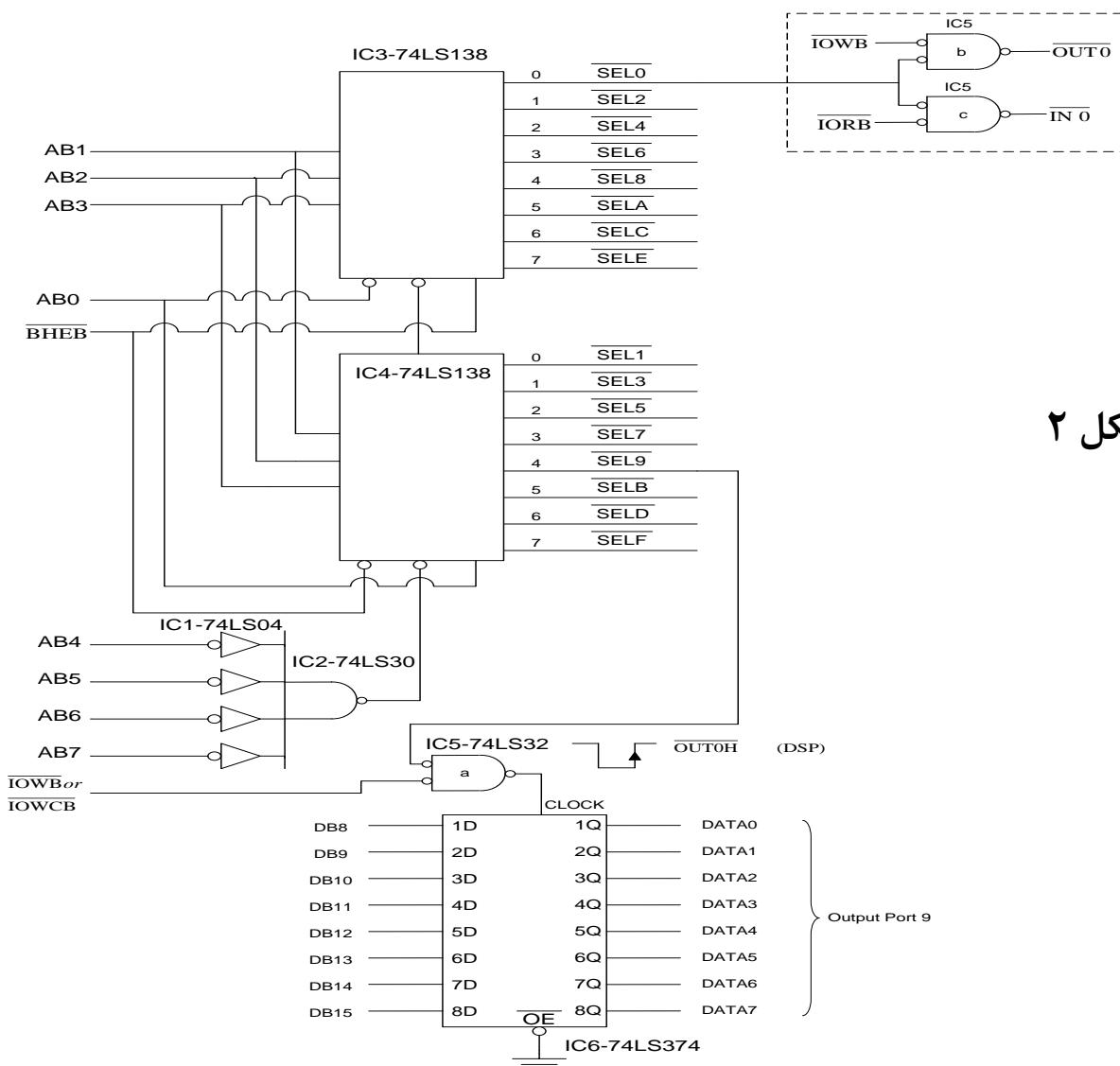
$$TWLWH = 2TCLCL - 60ns = 2 \times 200ns - 60ns = 340ns \text{ (8086)}$$

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

$$TWHDX = TCLCH - 30ns = 118ns - 30ns = 88ns \text{ (8086)}$$



طراحی پورت خروجی موازی (ادامه)



شکل ۲

طراحی پورت خروجی موازی(ادامه)

حتی با در نظر گرفتن مقدار ۰ 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} به وضعیت high برگشت، بر خطوط باس نگه داشته می‌شود. در این مورد و در بدترین حالت خواهیم داشت:

$$TWHDX_{min} = 88$$

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

که باز هم به سادگی قابل تحقق است.

$$\begin{aligned} t_h &\leq TWHDX_{min} - (t_{CBbufmax} + t_{IC5max} - t_{DBbufmin}) \\ &\leq 88 - (30 + 22 - 0) = 36ns \end{aligned}$$

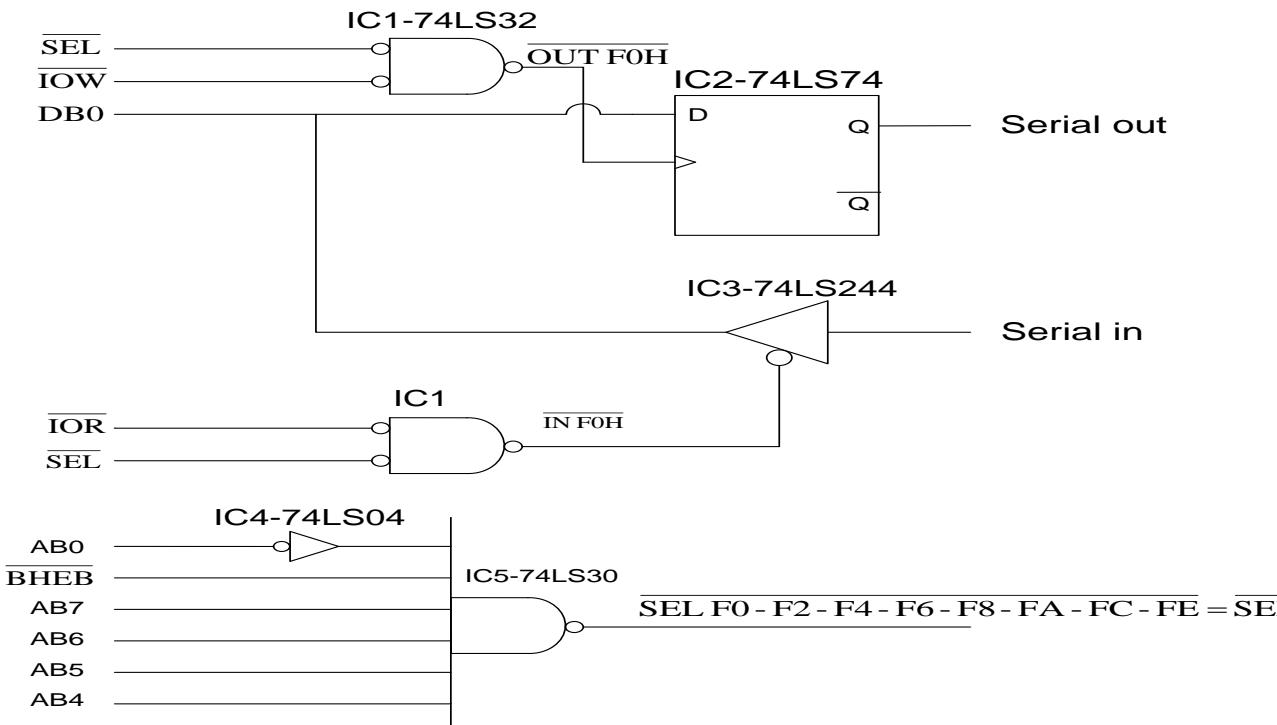
I/O سریال

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

دیکود جزئی برای انتخاب همهٔ پورت‌های بین XXFEH و XXF0H

سیگنال‌های انتخاب دستگاه (DSP)، داده‌ی DB0 را بر روی لچ قرار می‌دهند یا داده‌ی سریال ورودی را بر روی خط باس داده‌ی DB0 قرار می‌دهند.

لذا داده‌ی سریال به عنوان بیت صفر از پورت F0H (یا F4H، F2H، ...) 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 بیتهاي پایاني

- دادهای که می خواهیم بصورت ناهمگام ارسال شود، بین یک بیت شروع (همیشه بیت ۰) و یک یا دو بیت پایان (همیشه بیت ۱) قرار می گیرد.

- بیتهاي شروع و پایان حامل داده نیستند ولی به دلیل ماهیت آسنکرون انتقال، وجود آنها لازم است.

نرخ باودهای استاندارد
برای ارتباط دادهی سریال

75

نرخ باود

110

- نرخ داده را می توان بر حسب بیت بر ثانیه یا کاراکتر بر ثانیه بیان کرد.

150

300

Baud Rate

600

1200

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

2400

- بیتهاي داده به ازای هر کاراکتر؛ معمولاً بین ۵ تا ۸.

4800

- تعداد بیتهاي توقف، ۱ یا ۲.

9600

- بیت توازن برای تشخیص خطای تک بیتی؛ توازن می تواند فرد یا زوج باشد یا اصلاً به کار نرود.

19200

- نرخ باود.

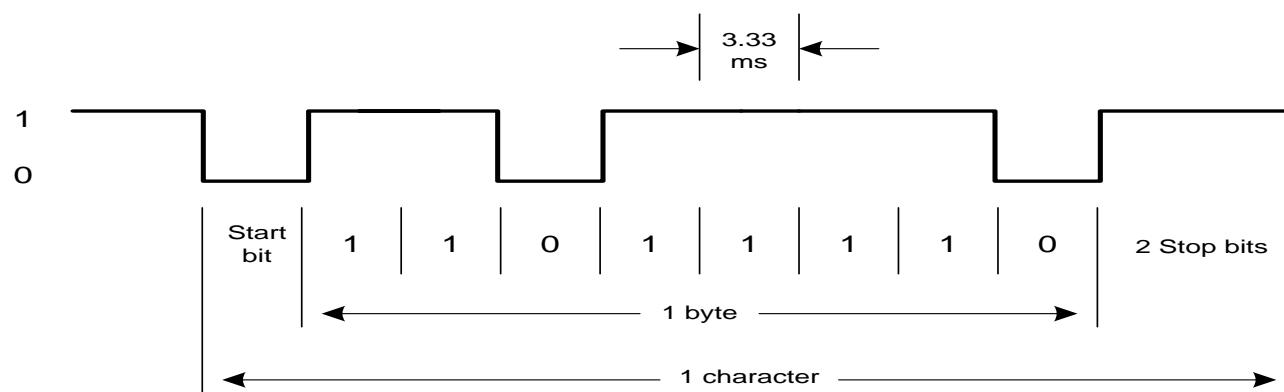
-
-
-

نرخ باود

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

حل: چون هر بیت به مدت ۳.۳۳ میلی ثانیه موجود است، نرخ انتقال بیت برابر با $300 = \frac{1}{3.33 \text{ ms}}$ ۱ بیت بر ثانیه 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 $27.3 = \frac{1}{36.63}$ ۱ کاراکتر بر ثانیه.



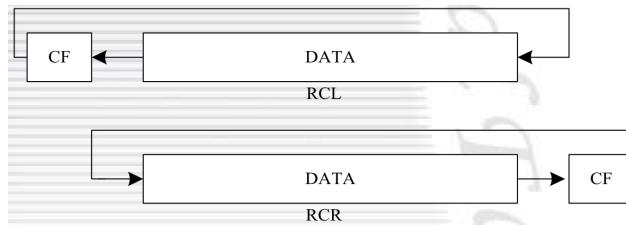
شکل ۷

تولید داده‌ی سریال ناهمگام

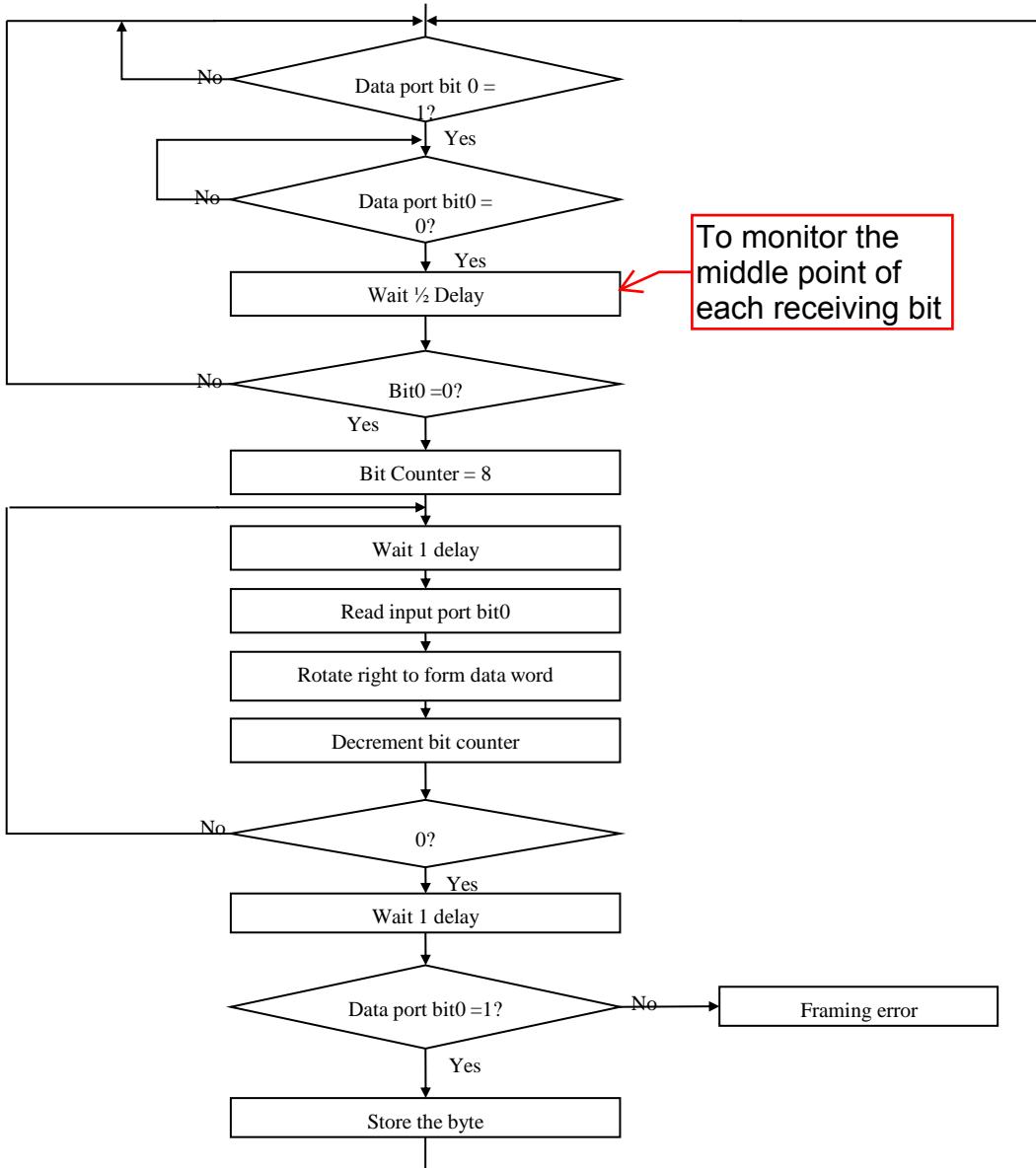
مثال: برنامه زیر یک بایت داده را در یک قاب قرار داده و ارسال می‌نماید:

- 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 DPORt	DELAY:NEAR EQU 0F0H
0000	CODE	SEGMENT ASSUME	CS:CODE
0000	PROC3	PROC	NEAR
0000	B9 000B	MOV	CX, 11 ;11 bits/char
0003	F8	CLC	;Start bit
0004	D0 D0	RCL	;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	



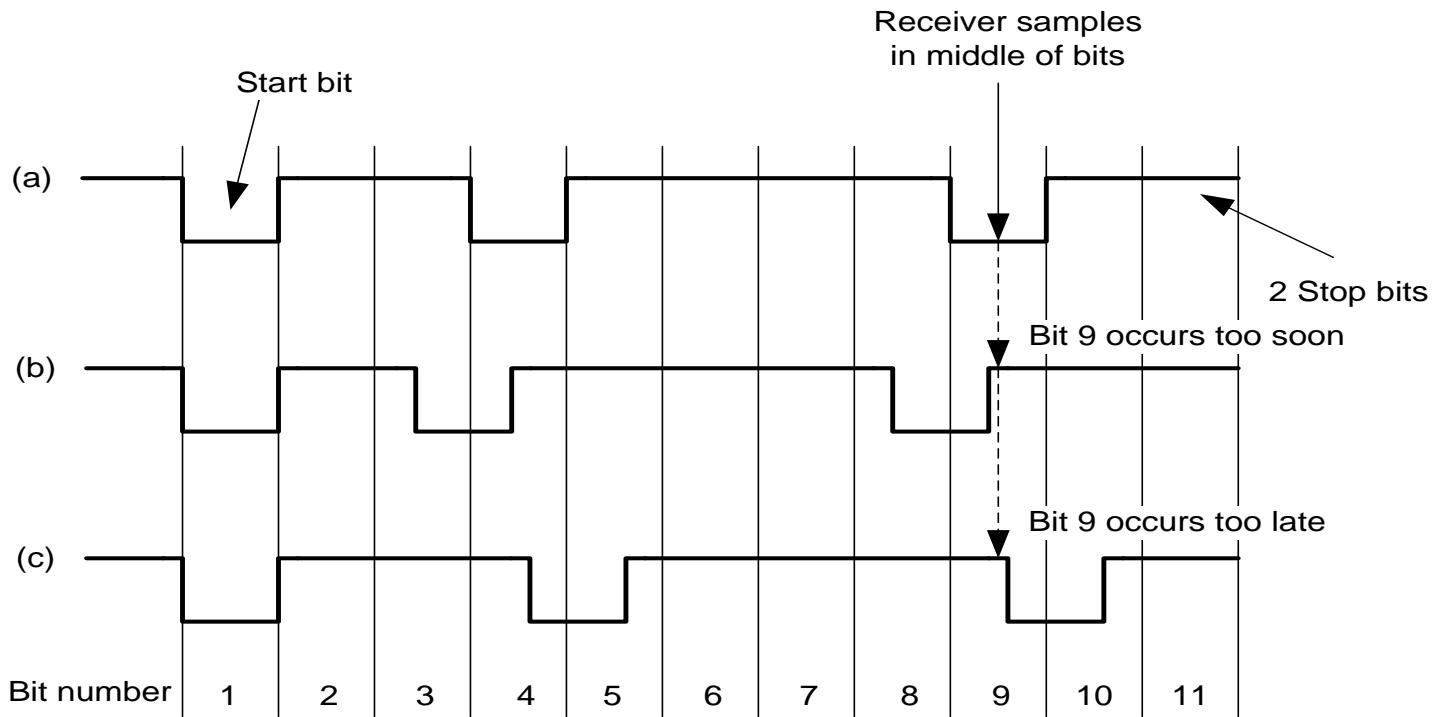
بازیابی داده‌ی سریال ناهمگام (ادامه)



فلوچارت ارتباط ناهمگام

شکل ۸

بازیابی داده‌ی سریال ناهمگام (ادامه)



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



I/O برنامه‌ریزی شده

- در روش سرکشی (Polling) 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 میلی ثانیه 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 برآکت 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 کدام هر کدام از سیم‌های سیگنال به کار می‌روند.

پین سیگنال بازگشت	پین	جهت سیگنال	توصیف
1	19	<u>STROBE</u>	In پالس STROBE برای خواندن داده‌ی ورودی. طول پالس در ترمینال دریافت باید بیش از ۰.۵ 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 ۱ در سطح HIGH و به ازای ۰ 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	<u>ACKNLG</u>	Out پالسی به طول تقریبی ۰.۵ 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 کدام هر کدام از سیم‌های سیگنال به کار می‌روند.

توصیف	جهت سیگنال	پین سیگنال	پین بازگشت	پین	1
پالس STROBE برای خواندن داده‌ی ورودی. طول پالس در ترمینال دریافت باید بیش از ۰.۵ میکروثانیه باشد. سطح سیگنال معمولا HIGH است و خواندن داده در سطح LOW این سیگنال انجام می‌شود.	In	<u>STROBE</u>	19	1	
این سیگنال‌ها به ترتیب اطلاعات اولین تا هشتمین بیت داده‌ی موازی را نشان می‌دهند. هر سیگنال به ازای ۱ در سطح HIGH و به ازای ۰ در سطح LOW قرار می‌گیرد.	In	DATA 1	20	2	
	In	DATA 2	21	3	
	In	DATA 3	22	4	
	In	DATA 4	23	5	
	In	DATA 5	24	6	
	In	DATA 6	25	7	
	In	DATA 7	26	8	
	In	DATA 8	27	9	
پالسی به طول تقریبی ۰.۵ میکروثانیه بیان می‌دارد که داده دریافت شده و چاپگر آماده‌ی دریافت داده‌ی بعدی است.	Out	<u>ACKNLG</u>	28	10	

چاپگر موازی

توصیف	پین سیگنال	پین بازگشت	سیگنال	جهت	
سطح HIGH بیان می کند که چاپگر نمی تواند داده ای دریافت کند که در موارد زیر اتفاق می افتد:	11	29	BUSY	Out	<ul style="list-style-type: none"> • در حین دریافت داده • در حین عملیات چاپ • در وضعیت OFF-LINE • در وضعیت خطای چاپگر
سطح HIGH بیان می کند که کاغذ چاپگر تمام شده است.	12	30	PE	Out	
این سیگنال بیان می کند که چاپگر در وضعیت خواسته شده است.	13	-	SLCT	Out	
قرار دادن این سیگنال در سطح LOW باعث می شود که بعد از چاپ به طور خودکار کاغذ یک خط به پایین رود.	14	-	<u>AUTO</u> <u>FEED XT</u>	In	
بدون استفاده	15	-	NC	-	
سطح منطقی زمین	16	-	0V	-	
زمین بدنه ی چاپگر که مستقل از سطح منطقی زمین است.	17	-	CHASIS-GND	-	
بدون استفاده	18	-	NC	-	
سیگنال بازگشت زوج های به هم تابیده شده که در سطح LOW قرار می گیرند.	19تا30	-	GND	-	

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: <ul style="list-style-type: none"> • وضعیت اتمام کاغذ • وضعیت OFF-LINE • وضعیت خطأ
33	-		GND	-	همانند پین‌های 19 تا 30
34	-		NC	-	بدون استفاده
35	-				از طریق مقاومت $4.7K\Omega$ 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 لج می‌کند.
- دو سیگنال کنترلی دیگر نیز برای چاپگر فراهم شده‌اند که BUSY و \overline{ACKNLG} نام‌گذاری شده‌اند. با توجه به جدول Y 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.
- 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".
- است ولی به جای یک پالس، یک سطح را فراهم می‌کند.

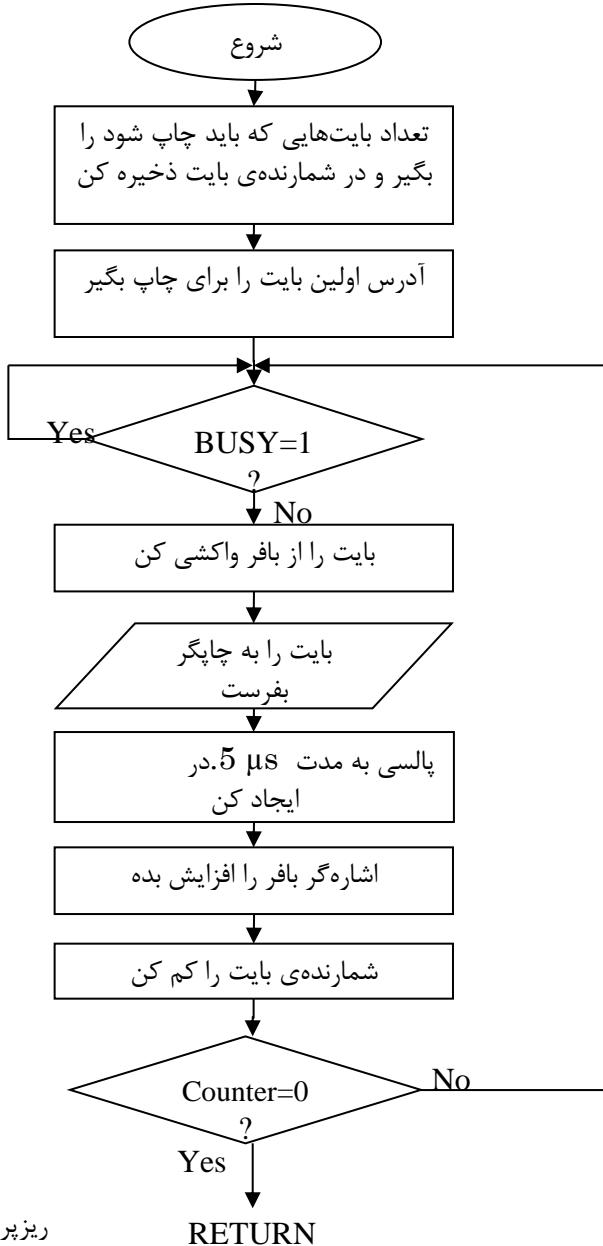
روش سرکشی (ادامه)

- 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 مرتباً 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 مجدداً در حلقه‌ی سرکشی قرار می‌گیرد.
- قبل از نوشتمن 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 لج هشت بیتی برای نگهداشتن داده در پین‌های ورودی چاپگر به کار رفته است. کلاک این لج از سیگنال IC2-a خروجی می‌آید. سیگنال‌های SEL2 و SEL3 از خروجی‌های دیکودر نشان داده شده در شکل ۲ می‌آیند.

روش سرکشی (ادامه)

شکل ۱۳

فلوچارت ارسال داده برای پرینتزر همراه با handshaking

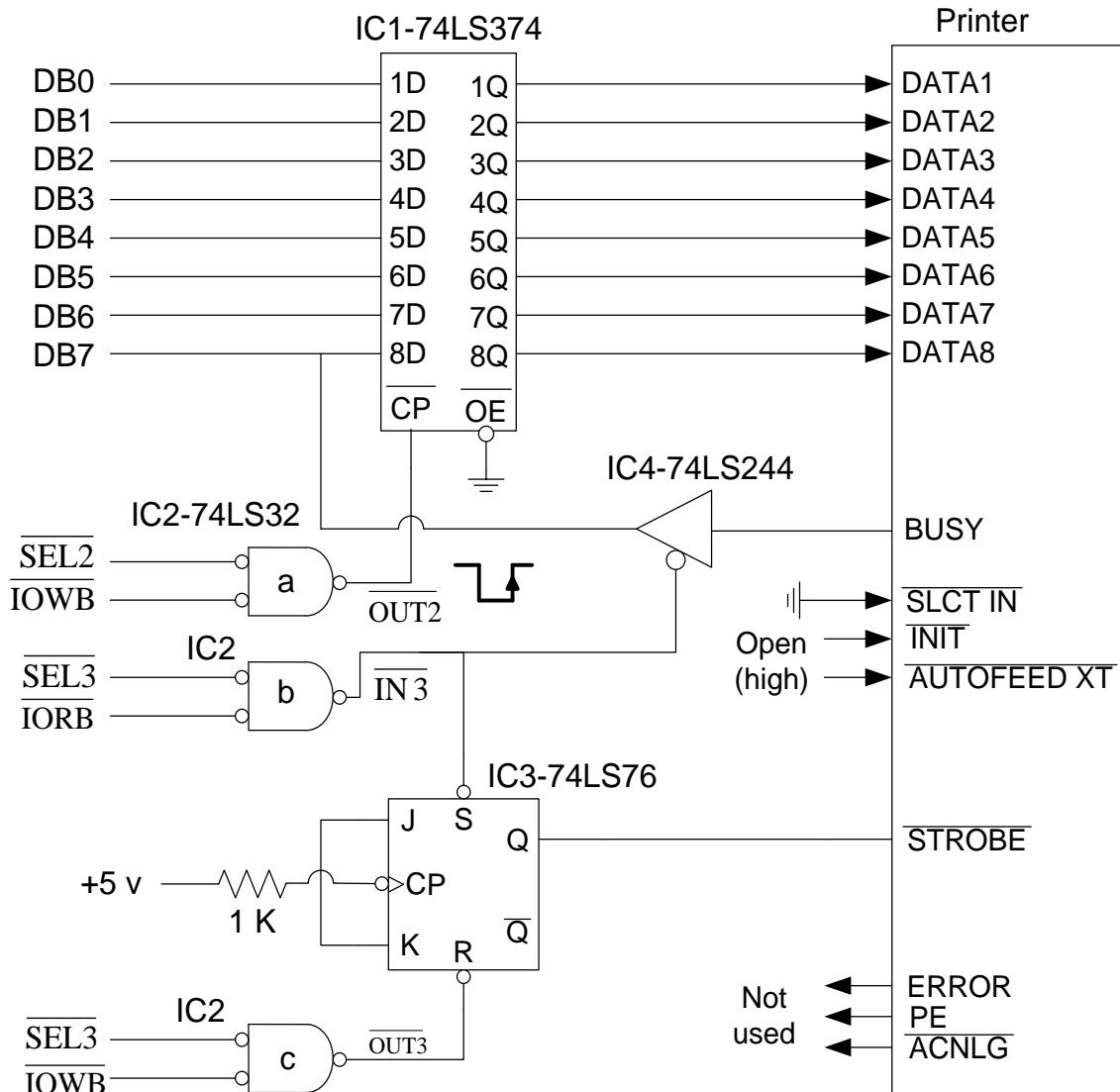


روش سرکشی (ادامه)

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

```
OUT 3, AL      ; STROBE = 0
IN AL, 3       ;
```
- چون دستور IN برای اجرا به ۱۰ کلاک نیاز دارد، با فرض کلاک ۵MHz، سیگنال $\overline{\text{STROBE}}$ به مدت $2\mu\text{s}$ 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 ۳ می‌خواند.
- دستور IN AL, BUSY پالس $\overline{\text{STROBE}}$ را غیرفعالی می‌کند. چون دستور LOOP POLL بین دستورات فعل 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 STROBE برای همگام کردن ریزپردازنده و دستگاه جانبی است. چون انتقال داده به چاپگر تحت کنترل نرمافزار I/O انجام می‌شود، این تکنیک را 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.

		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 برآکت بیانگر تعداد کلاک‌های لازم برای اجرای این دستورات است که جمua ۳۰ کلاک (۶μs برای کلاک ۵MHz) 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 $1666 = 10\text{ms} / 6\mu\text{s}$ 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 تحويل دادن هر کاراکتر به آن بیش از $10\mu\text{s}$ طول نمی‌کشد.
- در این مدت CPU دستور دیگری جز سرکشی بیت وضعیت چاپگر را انجام نمی‌دهد. اگر CPU کار 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 حداقل 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 (مجموعاً ۷۹ کلاک) که در یک سیستم با کلاک ۵MHz به $15.8\mu s$ زمان نیاز دارد و لذا نرخ ارسال داده ۶۳۲۹۱ کاراکتر بر ثانیه خواهد بود.



زمان پاسخ‌دهی

- یک سیستم مبتنی بر ریزپردازنده نوعاً 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.
- | Video Terminal | Modem | DAC | ADC | Plotter | Daisy Wheel Printer | Line Printer | Floppy disk |
|----------------|-------|-----|-----|---------|---------------------|--------------|-------------|
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

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 CPU به آن سرویس‌دهی می‌کند. در مثال قبل چون روتین حلقه‌ی سرکشی $6\mu s$ 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 کاملاً 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.



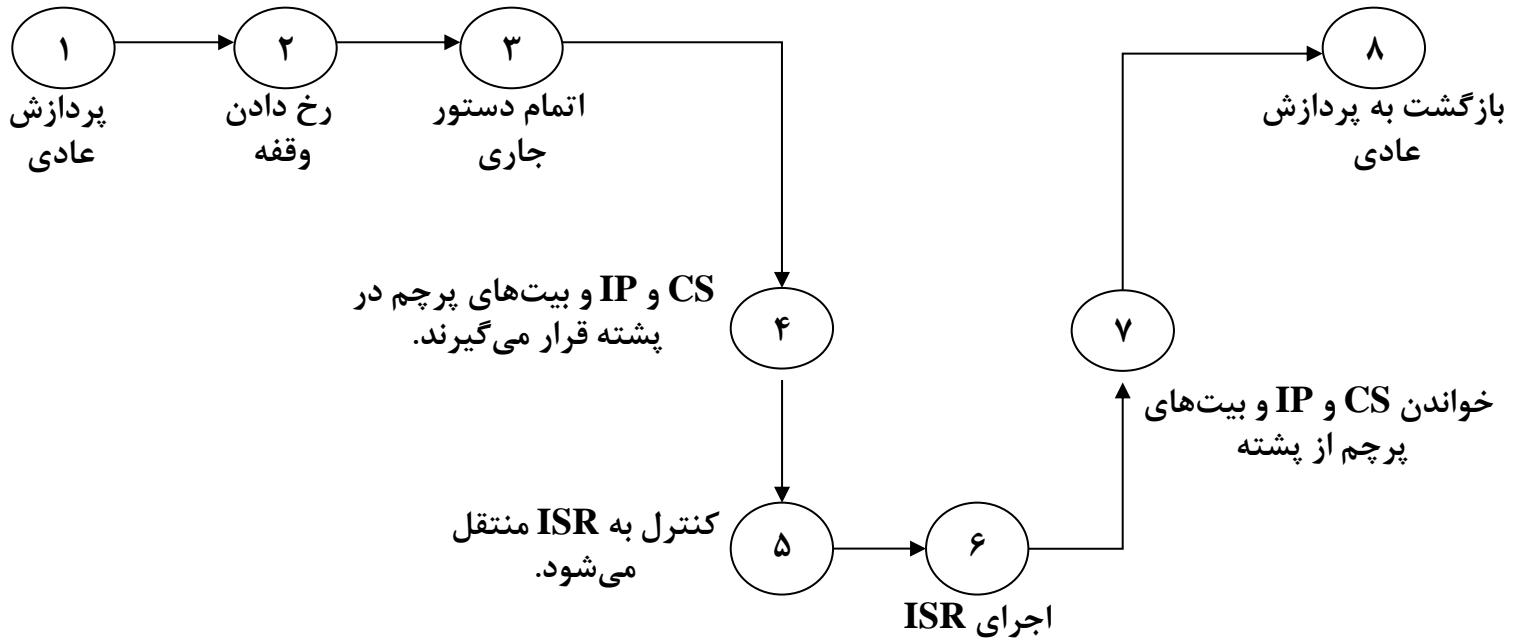
I/O وقفه‌گرا (ادامه)

- شکل ۱۶ پاسخ CPU به یک وقفه را نشان می‌دهد.
- در مرحله 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 مرحله ۳، محتوای رجیسترهاي IP و بیت‌های پرچم در پشته قرار می‌گیرند (مرحله ۴)، و مرحله ۵ انتقال کنترل برنامه به ISR است.
- بعد از اجرای روتین وقفه در مرحله ۶، محتوای پشته که رجیسترهاي IP,CS و بیت‌های پرچم است از پشته بازیابی می‌شود و مرحله ۷ به اتمام می‌رسد. در مرحله ۸ پردازش عادی CPU ادامه می‌یابد.
- اگر فرض کنیم $100\mu s$ 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 ($10000\mu s = 1/100$) باشد آنگاه $9900\mu s$ 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 رخدادن وقفه



انواع وقفه

- ۸۰۸۶ دارای هفت نوع وقفه‌ی متفاوت می‌باشد که در جدول ۴ آورده شده‌اند. INTR و NMI وقفه‌های خارجی هستند که از طرف سخت‌افزار اعمال می‌شوند. n INT، 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 دستور العمل جاری قبل از پاسخگویی به وقفه به اتمام می‌رسد.
- وقفه‌های داخلی به جز وقفه‌ی تک گامی نسبت به وقفه‌های خارجی اولویت دارند.
- اگر همزمان وقفه‌ایی بر NMI و INTR اتفاق افتد، ابتدا وقفه‌ی NMI پاسخ داده می‌شود.

جدول ۴

انواع وقفه‌های ۸۰۸۶

نام وقفه	نحوه مقداردهی	قابل پوشش؟	چگونگی تحریک	اولویت	سیگنال Acknowledge	آدرس جدول بردار	تا خیر وقفه
NMI	سخت افزار خارجی	خیر	لبه‌ی پالس، حداقل ۲ پالس کلاک	2	ندارد	00008H-0000BH	دستور جاری + کلاک ۵۱
INTR	سخت افزار خارجی	بلی از طریق IF	سطح بالا تا زمانیکه تایید شود	3	بلی از طریق IF	سخت افزار خارجی	دستور جاری + کلاک ۶۱
INT n	داخلی، نرم‌افزاری	خیر	ندارد	1	ندارد	N * ۴	کلاک ۵۱
INT 3	داخلی، نرم‌افزاری	خیر	ندارد	1	ندارد	0000CH-0000FH	کلاک ۵۲
INTO	داخلی، نرم‌افزاری	بلی از طریق OF	ندارد	1	ندارد	00010H-00013H	کلاک ۵۳
تقسیم بر 0	CPU داخلی	خیر	ندارد	1	ندارد	00000H-00003H	کلاک ۵۱
تک گامی	CPU داخلی	بلی از طریق TF	ندارد	4	ندارد	00004H-00007H	کلاک ۵۱

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

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

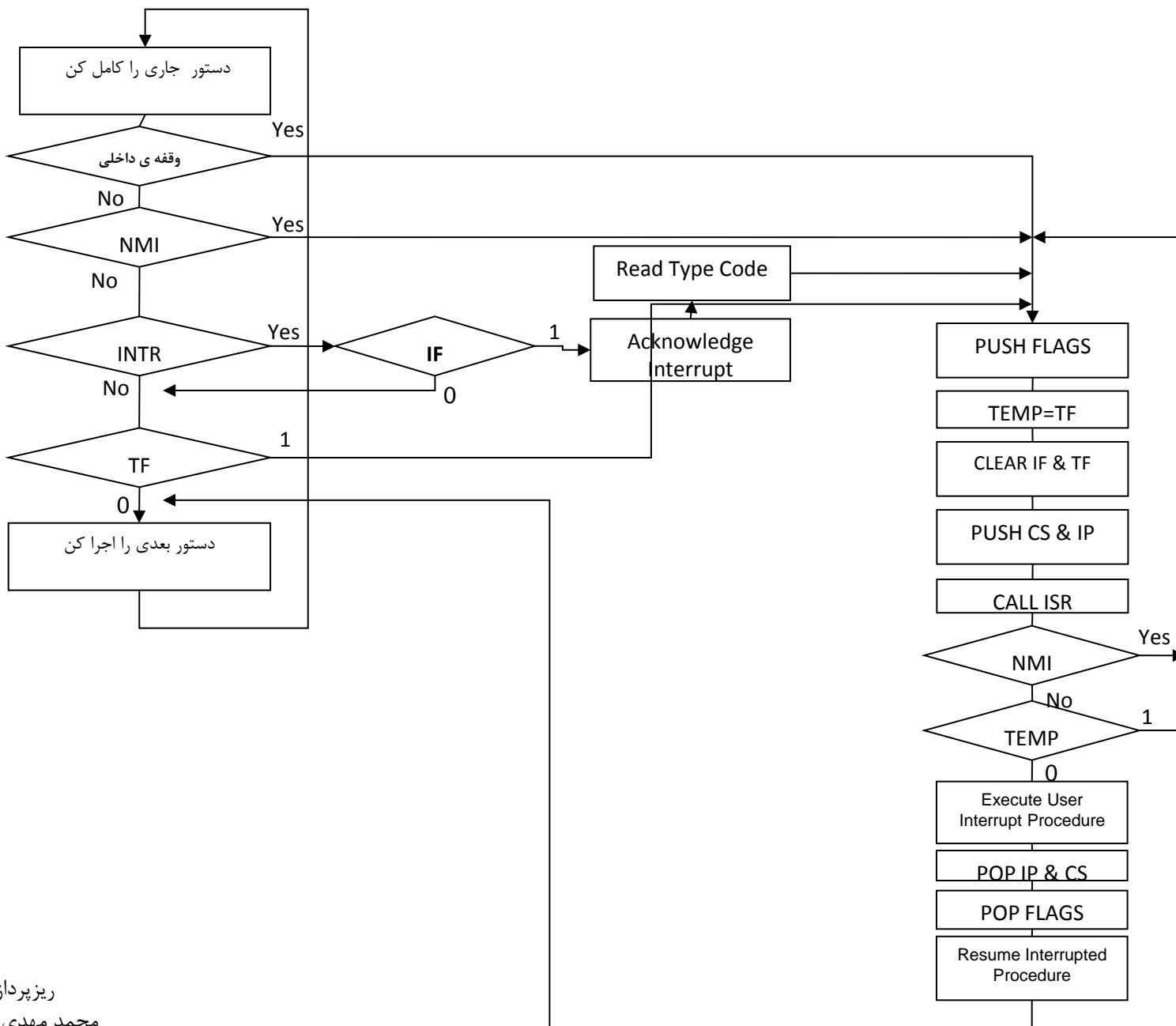
چنانچه دستوری بنام INT اجرا شود وقفه نوع ۳ یعنی INT 3 اجرا می‌شود.

أنواع وقفه های ۸۰۸۶

- چنانچه تقسیم بر صفر رخ دهد وقفه نوع ۰ یعنی وقف ۰ INT رخ می‌دهد.
- پردازش تک گام وقفه نوع ۱ است.
- چنانچه پایه NMI فعال شود وقفه نوع ۲ یعنی ۲ INT اجراء می‌شود.
- چنانچه دستوری بنام INT اجرا شود وقفه نوع ۳ یعنی ۳ INT اجرا می‌شود.
- وقفه نوع ۴ است یعنی INT4 اجرا می‌شود.



دبالهی پردازش وقفه



انواع وقفه (ادامه)

- وقتی یک وقفه پاسخ داده می‌شود، محتوای رجیسترهاي IP و CS در پشته قرار می‌گيرند و لذا شش بايت در پشته قرار می‌گيرند.
- اگر پرچم TF=1 باشد، بعد از اين فرآيند همچنان در وضعیت تک گامی می‌ماند.
- به هر حال 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 أولویت 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 افتاد 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.
- 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 نوشتan 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 مدت 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: INTR، NMI و تک گامی. بالاترین اولویت به NMI اختصاص دارد و لذا ابتدا این وقفه تشخیص داده می‌شود. ولی به هر حال بعد از قرار دادن CS، IP و بیت‌های پرچم در پشتی وقفه‌ی تک گامی تشخیص داده می‌شود (چون $TEMP=1$ است).

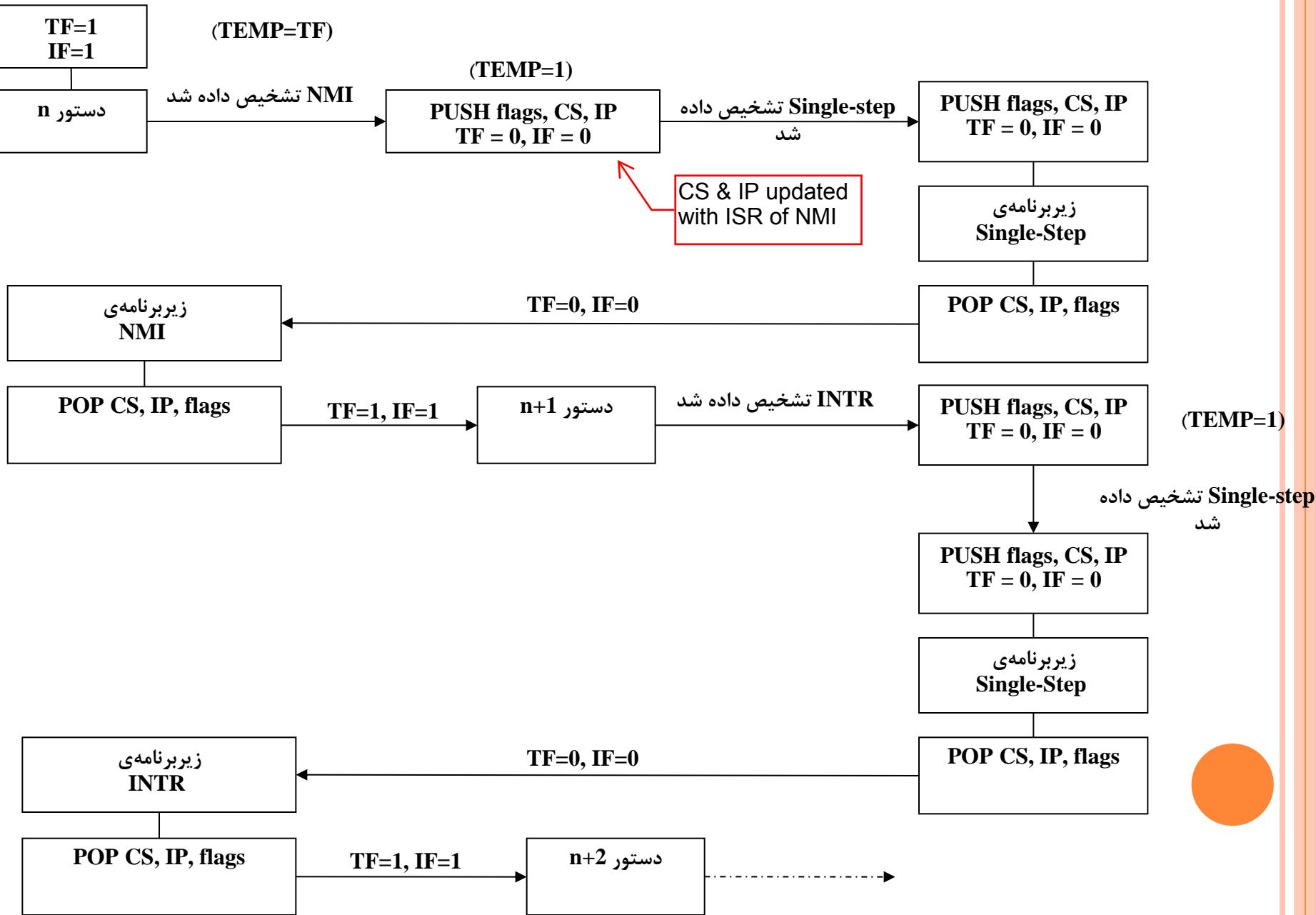
در واقع زیربرنامه تک گامی شدن دستورالعمل جاری قبل از روتین وقفه‌ی NMI فراخوانی می‌شود

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 پسته قرار می‌گیرند و $TF=0$ می‌شوند ولی مجدداً 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 مجدداً چون $TF=0$ است این روتین در مد تک‌گامی 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 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 H00000 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 بایتی حاصل به یکی از ۲۵۶ بردار چهار بایتی اشاره می‌کند.



أنواع وقفه (ادامه)

مثال: عدد نوع وقفه خاصی برابر $n=41H$ است. اگر آدرس روتین وقفه 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 IP $00104H$ در آدرس کم ارزش و CS در آدرس پرارزش قرار می‌گیرد.

Address	Content Value
00107H:	09H
00106H:	E3H
00105H:	00H
00104H:	10H

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

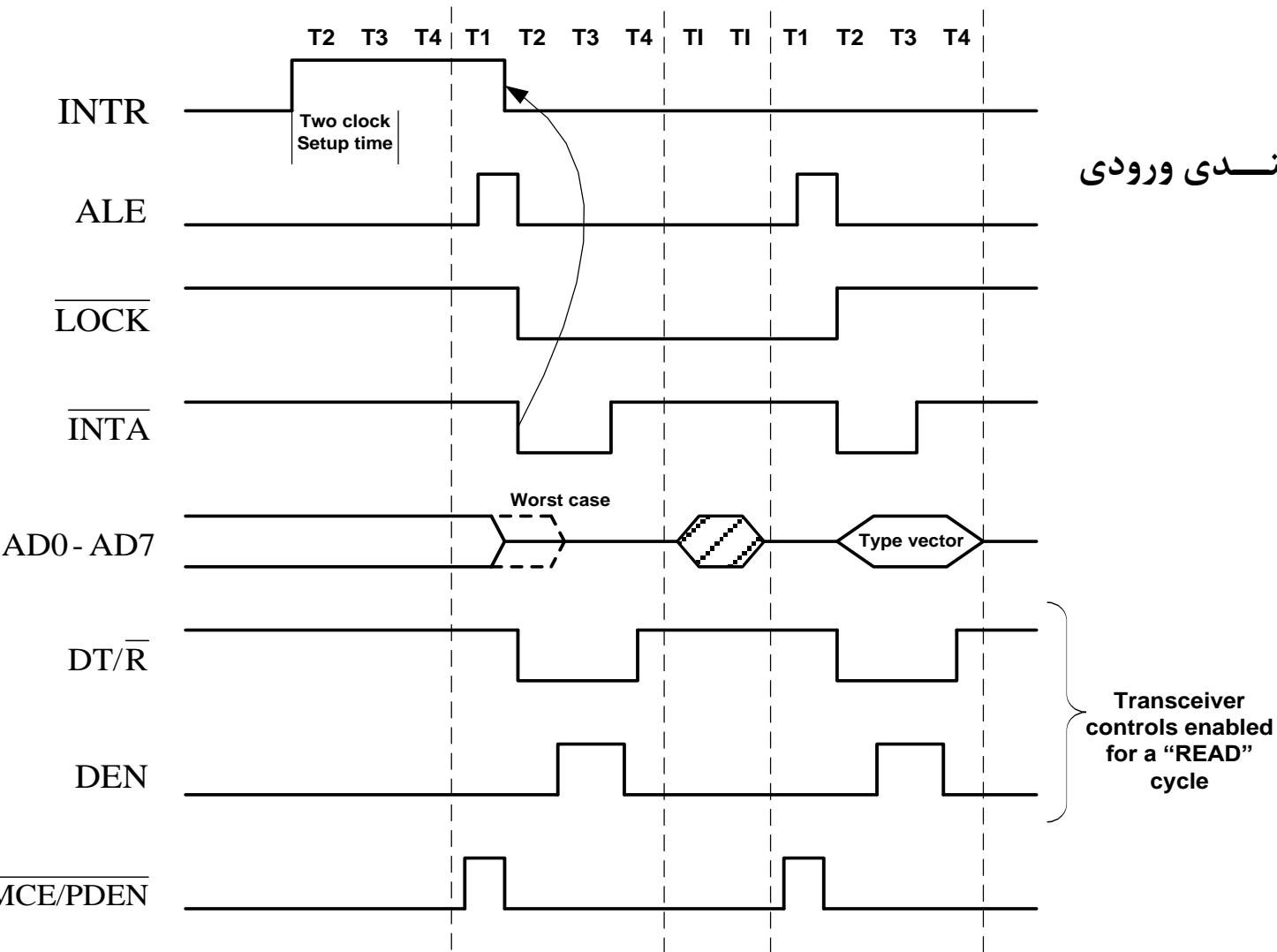
زمانبندی وقفه‌ی خارجی

- ۸۰۸۶ ورودی‌های INTR و NMI را در آخرین کلاک از آخرين سیکل باس هر دستورالعمل نمونه‌برداری می‌کند.
- حساس به لبه‌ی بالا رونده است و به طور داخلی 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 زمانی که تشخیص آن از طرف CPU اعلام شود.
- سیگنال خروجی CPU برای اعلام تشخیص وقفه است. وقتی $\overline{\text{INTA}} = 0$ است ۸۰۸۶ وقفه را تشخیص داده و پذیرفته است و آن هم زمانی رخ می‌دهد که $\text{IF} = 1$ باشد.
- سیگنال $\overline{\text{INTA}}$ تنها برای وقفه‌ی INTR استفاده می‌شود و برای دیگر وقفه‌های داخلی و نیز NMI اعلامی صورت نمی‌گیرد.

زمانبندی وقفه‌ی خارجی (ادامه)

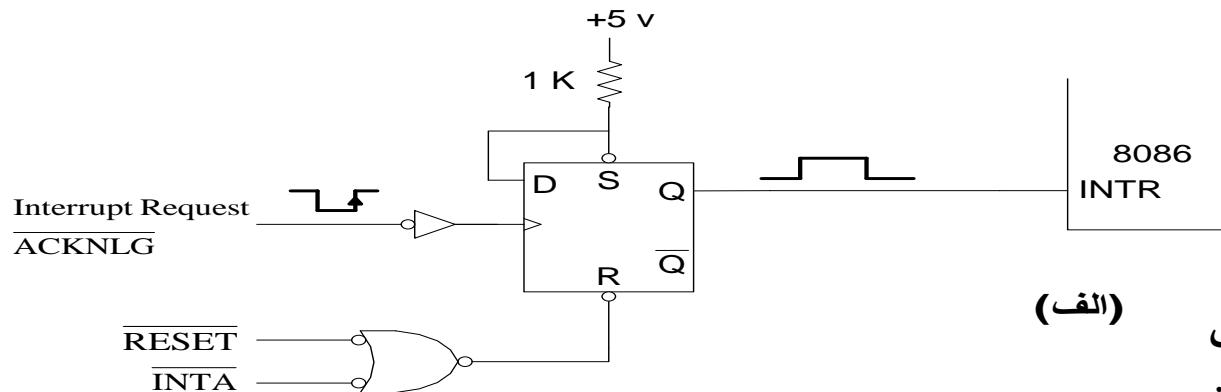
- شکل ۱۹ زمانبندی سیکل باس اعلام وقفه را نشان می‌دهد.
- INTR باید قبل از کلак T4 از دستورالعمل وقفه یافته، زمان راهاندازی به مدت دو کلак را برآورده کند.
- اگر این شرایط برقرار نباشد، وقفه پذیرفته نمی‌شود تا پایان دستورالعمل بعدی.
- نکته اینکه این زمان انتظار ممکن است بیش از ۱۰۰ کلak به طول انجامد که این مساله در مورد دستورات ضرب و تقسیم رخ می‌دهد.
- بعد از پذیرفته شدن وقفه دو سیکل باس اجراء می‌شود که با دو سیکل باس بیکار از هم جدا شده‌اند.
- اولین پالس، درخواست وقفه را اعلام 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)



شکل ۱۹- زمانبندی ورودی INTR

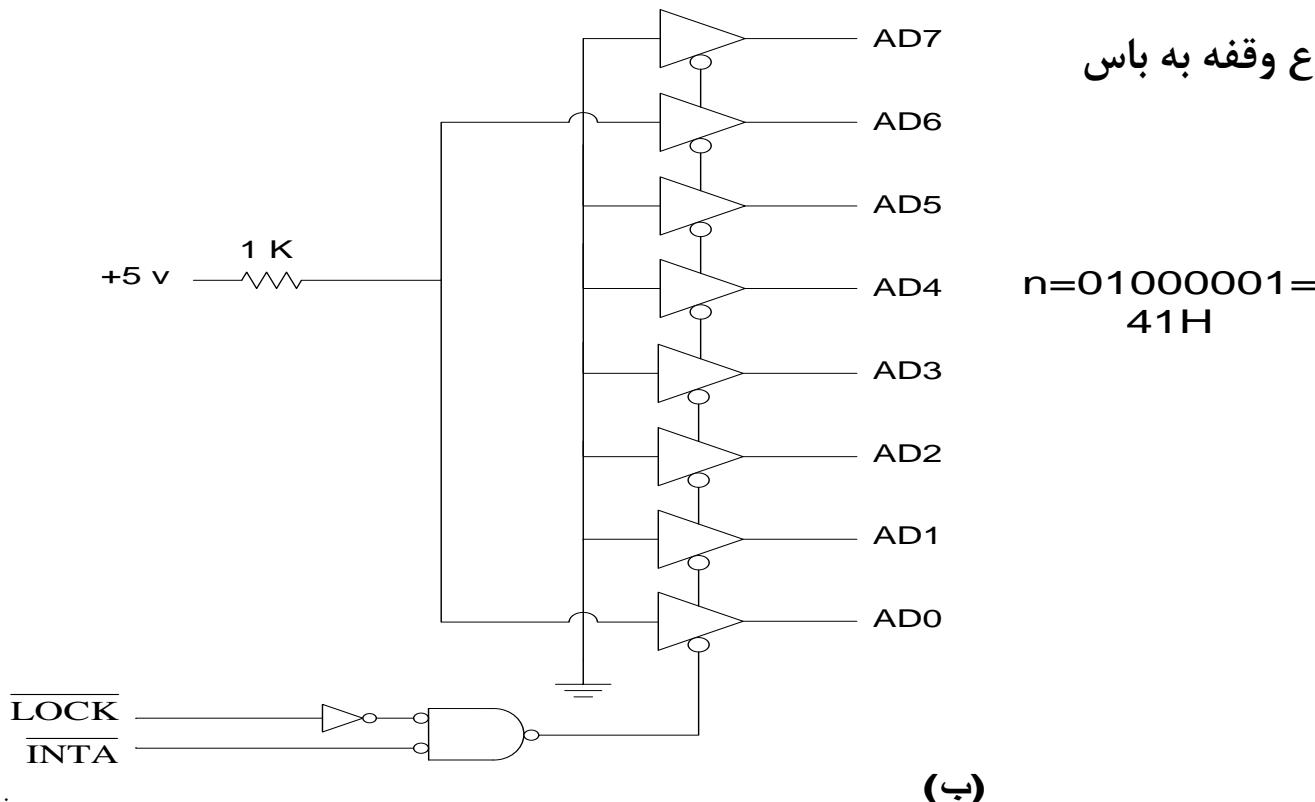
زمانبندی وقفه‌ی خارجی (ادامه)



شکل ۲۰

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

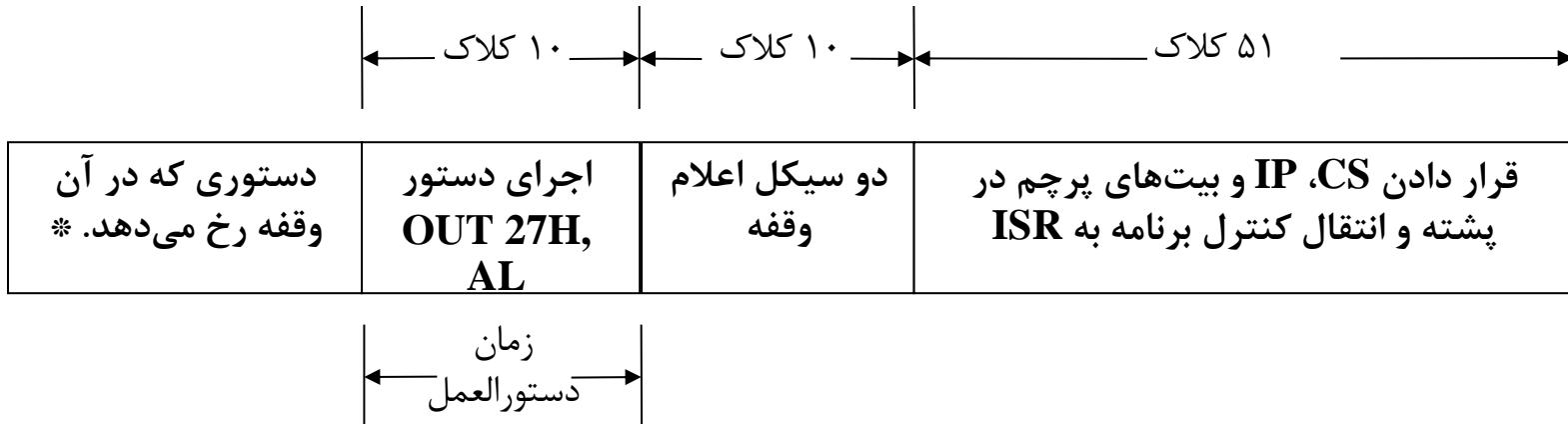
ب: تحویل عدد نوع وقفه به باس داده



زمان پاسخگویی

- حلقه‌ی اصلی روش سرکشی به ۳۰ کلک زمان برای اجرا نیاز داشت. لذا پاسخگویی به هر دستگاه جانبی $6\mu s$ 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 اعلام وقفه
 - قرار دادن محتوای IP، CS و بیت‌های پرچم در پشتہ ISR
 - 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 کلک برای اجرای دستور جاری منتظر بماند، ۱۰ کلک دیگر برای سیکل‌های اعلام وقفه و ۵۱ سیکل دیگر هم برای قرار دادن IP و بیت‌های پرچم در پشتہ صرف می‌شود.
- این زمان در یک سیستم با کلک ۵MHz برابر با $14.2\mu s$ خواهد بود.

زمان پاسخگویی (ادامه)



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

$$= \frac{1}{f_{clock}} * (61 + T_{instruction})$$



زمان پاسخگویی (ادامه)

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

مد آدرس‌دهی پایه به همراه شاخص به ۱۲ کلک برای محاسبه‌ی آدرس موثر (EA) نیاز دارد. و به ازای CL=255 جمعاً ۱۰۵۲ کلک مورد نیاز است و زمان پاسخ‌گویی $210.4\mu s$ خواهد شد.

وقفه‌ی NMI سیکل‌های اعلام وقفه را انجام نمی‌دهد و لذا تاخیر آن ۱۰ کلک کمتر خواهد بود و لذا در مورد قبلی زمان پاسخگویی به $208.4\mu s$ 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 $210.4\mu s$ 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:
POP segment register یا **MOV segment register**
- وقفه‌ای که در حین اجرای این دستورات اتفاق افتاد 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.



راه اندازی بردار وقفه و روتین سرویس دهی وقفه (ادامه)

003A		COM_PROC:	PROC	NEAR	
003A	80 3E 0000 R 00		CMP	STATUS, GOOD	;Make sure previous print complete, else let STATUS = ERROR and quit.
003F	74 04		JE	SKIP	
0041	C6 06 0000 R 01		MOV	STATUS, ERROR	
0046	EB 14		JMP	SHORT QUIT	
0048	C7 06 0007 R 0000	SKIP:	MOV	CHAR_XFER, 0	;Reset characters transferred to 0.
004E	C7 06 0005 R 0200		MOV	CHAR_COUNT, 512	This example prints 512 bytes buffers, update STATUS to print-in-progress. Start the first byte "manually". Remaining bytes will print "automatically" so return to main program.
0054	C6 06 0000 R 02		MOV	STATUS, PIP	
0059	CD 41		INT	65	
005B	FB		STI		
005C	C3	QUIT:	RET		
005D		COM_PROC:	ENDP		

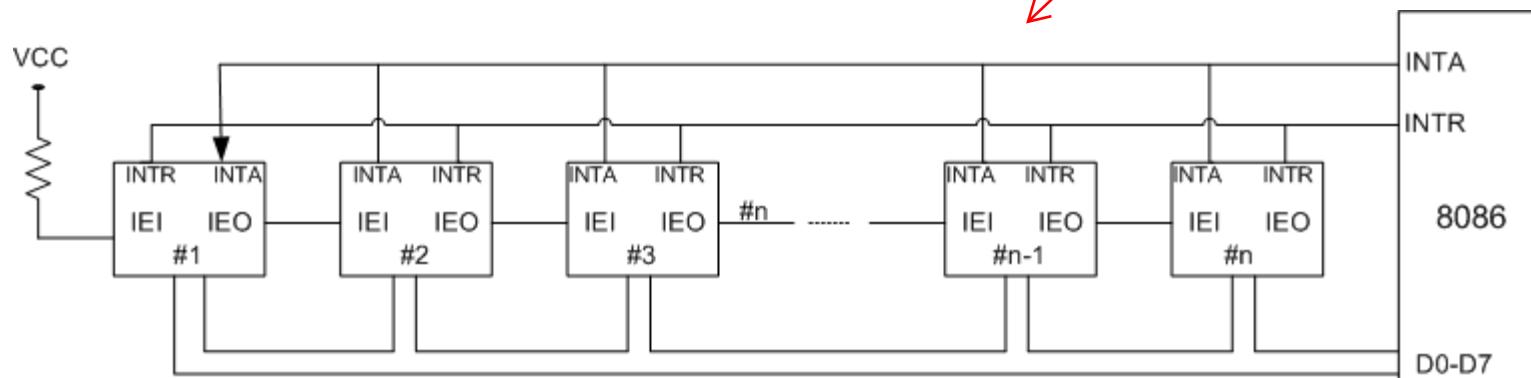


اولویتدهی وقفه‌ها

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

1. استفاده از **encoder**

2. روش **daisy Chain**



IEI: Interrupt Enable Input

IEO: Interrupt Enable Output

راه اندازی بردار وقفه و روتین سرویس دهی وقفه (ادامه)

در واقع مزیت روش مبتنی بر وقفه در این است که CPU تنها زمانی به سرویس دهی می پردازد که دستگاه جانبی آماده باشد و لذا در باقی زمان ها می تواند به اجرای دستورات برنامه ای اصلی خود بپردازد.

نرم افزار اضافی لازم در مد وقفه عیب اصلی این روش است. هر بار که ریز پردازنده به روتین ISR وارد می شود لازم است که آدرس بافر استخراج شود، کاراکتر واکشی و چاپ شود و شمارندهی تعداد کاراکترهای چاپ شده افزایش یابد. علاوه بر این هر رجیستری که در روتین وقفه استفاده می شود باید در پشتہ قرار گیرد.

برنامه ای که از مد سرکشی استفاده می کند هم باید همان متغیرها را واکشی کند ولی تنها یک بار در ابتدای برنامه لازم است.

اثر خالص آن افزایش تاخیر وقفه است (تا اجرای دستور چاپ) و کم شده نرخ عملیات چاپ و در نتیجه بعضی دستگاه های سریع که با مدهای سرکشی و DMA قابل کنترل هستند، را نمی توان با وقفه کنترل کرد.