

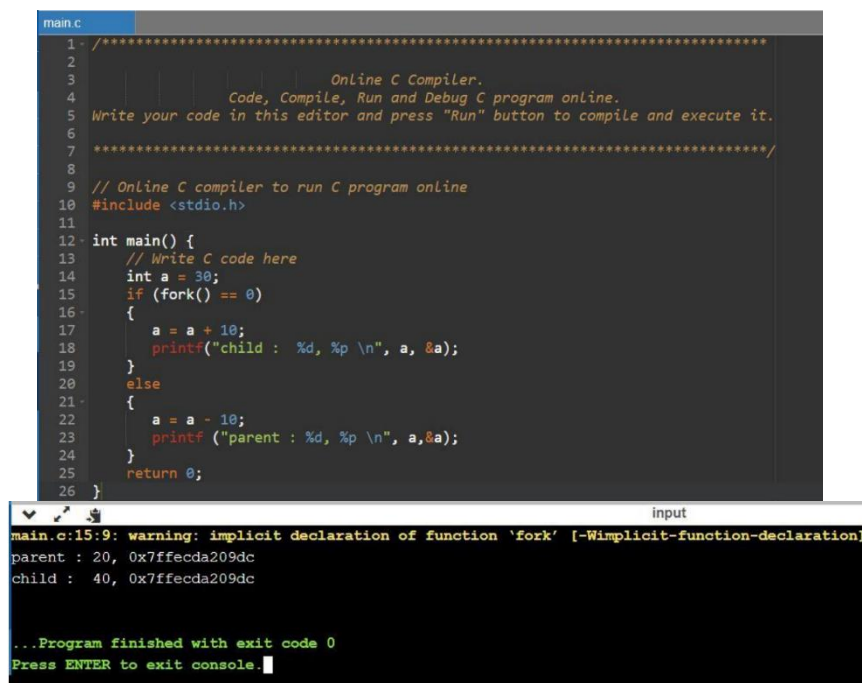
## سوال (۱)

هنگامی که فرایند فرزند ایجاد می شود ، از بخش های heap , stack و data فرایند والد یک کپی گرفته و اجرا می شود. با این حساب آدرس فیزیکی متغیر a در فرایند فرزند با پدر متفاوت خواهد بود.

اما در اینجا دو نکته مهم وجود دارد که باید به آن توجه کنیم:

نکته ۱: اینکه اول قسمت کد والد اجرا خواهد شد یا فرزند را نمی توان به صورت قطعی از قبل تعیین کرد. و لذا ممکن است ترتیب چاپ شدن خطوط فرق کند.

نکته ۲: علی رقم آنکه آدرس فیزیکی متغیر a در دو فرایند فرق می کند ، اما اگر این قطعه کد را در زبان C پیاده کنیم خواهیم دید که آدرس متغیر های a در هر دو فرایند یکسان خواهد بود. تصویر زیر نشان دهنده این مورد است:



```
main.c
1 //*****
2
3 Online C Compiler.
4 Code, Compile, Run and Debug C program online.
5 Write your code in this editor and press "Run" button to compile and execute it.
6 *****/
7
8 // Online C compiler to run C program online
9 #include <stdio.h>
10
11 int main() {
12     // Write C code here
13     int a = 30;
14     if (fork() == 0)
15     {
16         a = a + 10;
17         printf("child : %d, %p \n", a, &a);
18     }
19     else
20     {
21         a = a - 10;
22         printf("parent : %d, %p \n", a, &a);
23     }
24     return 0;
25 }
26

input
main.c:15:9: warning: implicit declaration of function 'fork' [-Wimplicit-function-declaration]
parent : 20, 0x7ffecda209dc
child : 40, 0x7ffecda209dc

...Program finished with exit code 0
Press ENTER to exit console.
```

علت این اتفاق در این است که سیستم های امروزی بجای آنکه با آدرس های physical متغیر ها کار کند ، با آدرس های logical آنها کار می کند. این موضوع باعث می شود که هنگام باز سازی آدرس متغیر ها مقادیر یکسانی ایجاد شود. لذا خروجی به صورت زیر است:

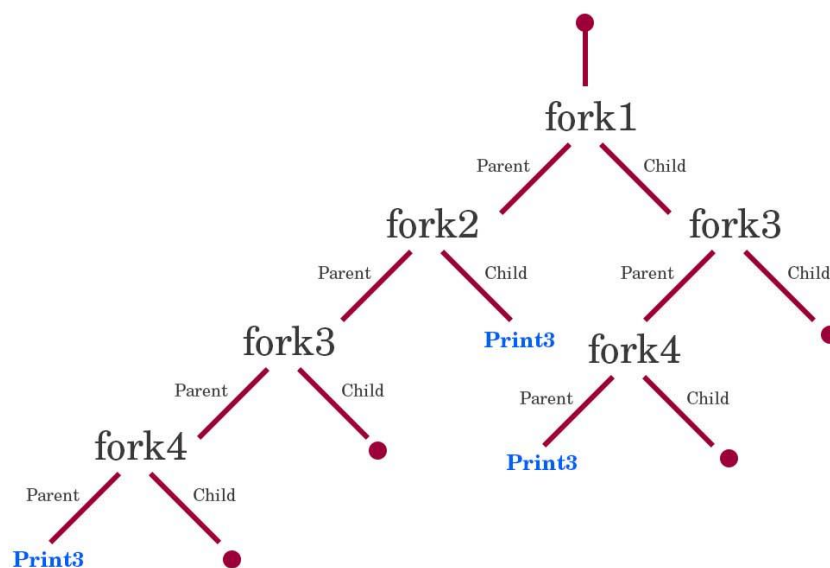
parent : "20, 2000"

child : "40, 2000"

### سوال ۲)

اگر fork ها را به ترتیب از چپ به راست شماره گذاری کنیم روند اجرای کد به صورت زیر است:

نکته مهمی که در اجرای روند کد وجود دارد مبحث Lazy Evaluation در زبان C است. به این صورت که اگر در سمت چپ یک عبارت And مقدار False ظاهر شود، سمت راست آن هر مقداری باشد حاصل And مقدار False خواهد بود. پس با ظاهر شدن مقدار False یا همان صفر در سمت چپ این عملگر، دیگر نیازی به اجرا و بررسی عبارت سمت راست نداریم. همین مبحث در مورد عملگر Or در صورتی که سمت چپ عملگر مقدار True ظاهر شود نیز وجود دارد.



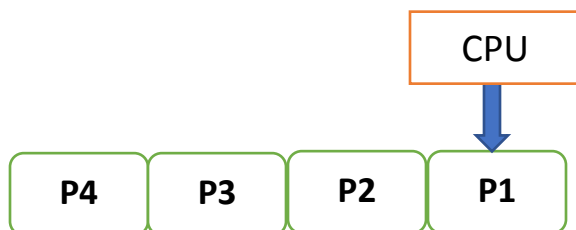
### سوال ۳)

پردازه zombie : پردازه ایی است که وظایف خود را به طور کامل انجام داده و زمان kill شدن آن فرا رسیده اما به دلایل مختلف مانند عدم ارسال سیگنال SIGCHLD توسط این پردازه (child) یا عدم اجرای wait() توسط پردازه پدر (parent) هنوز در process table باقی مانده است. روش handle کردن پردازه zombie : با فرستادن سیگنال kill نمیتوان این نوع پردازه ها را از بین برد چون به هیچ سیگنالی واکنش نشان نمیدهند (فقط توصیف گر آن در حافظه حضور دارد بنابراین اگر آن ها را هندل نکنیم با انباشته شدن سیستم را مختل میکنند) اما میتوان به پردازه پدر سیگنال SIGCHLD را ارسال کرد تا wait() را اجرا کند. اگر در پردازه پدر به درستی این امر پیاده سازی نشده باشد و قادر به اجرای wait() نباشد خود پردازه پدر را kill میکنیم تا پردازه zombie به پردازه orphan تبدیل شود.

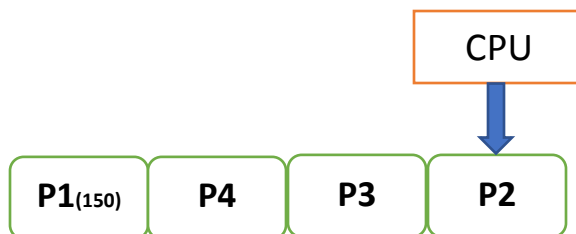
پردازه orphan : پردازه ایی است که قبل از اتمام وظایف خود به هر دلیلی پردازه پدر خود را از دست میدهد و پردازه پدرش بدون اجرای wait() برای این پردازه به پایان میرسد.

روش handle کردن پردازه orphan : پس از مدت کوتاهی که این پردازه orphan میشود پردازه init (اولین پردازه ایی که از زمان شروع به کار سیستم فعال میشود و تا زمان خاموش شدن آن از بین نمیرود با pid برابر با ۱) را به عنوان پدر این نوع پردازه ها اختصاص میدهیم. این پردازه در دوره های زمانی مشخص wait() را برای این نوع پردازه ها صدا میزند تا terminate شوند.

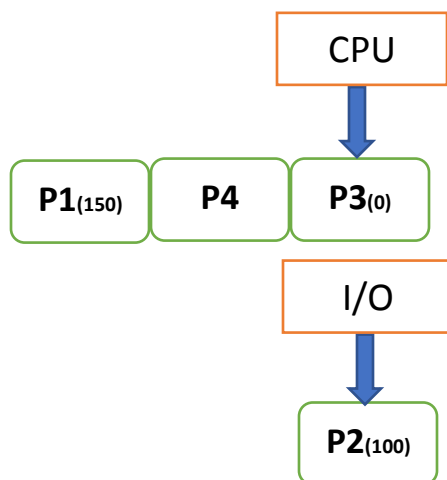
**T = 0**



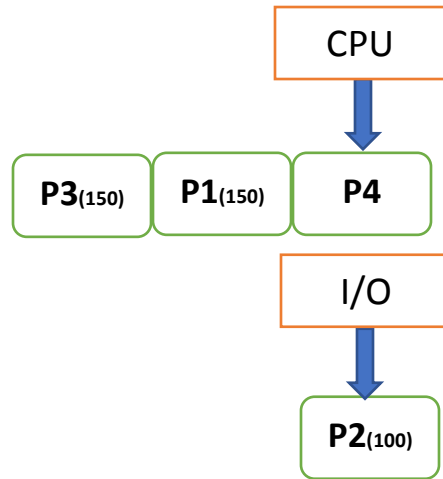
**T = 150**



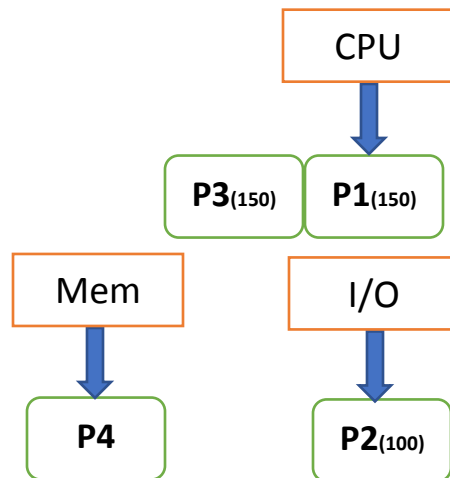
**T = 250**



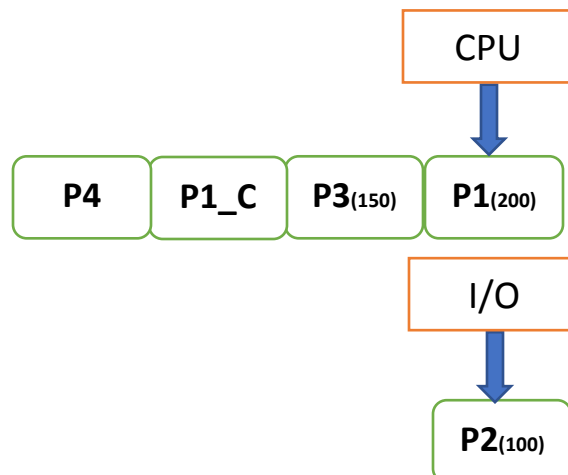
**T = 400**



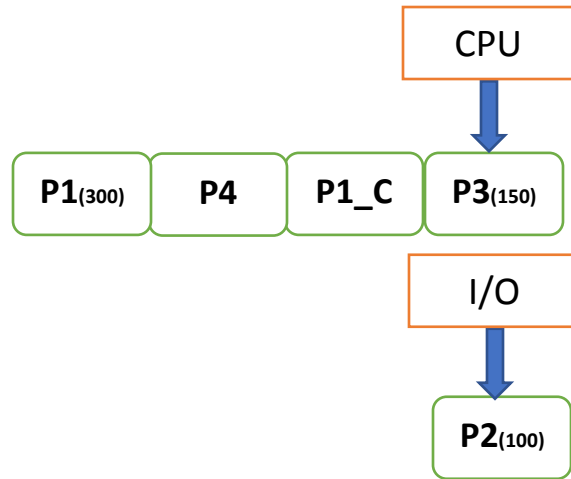
**T = 450**



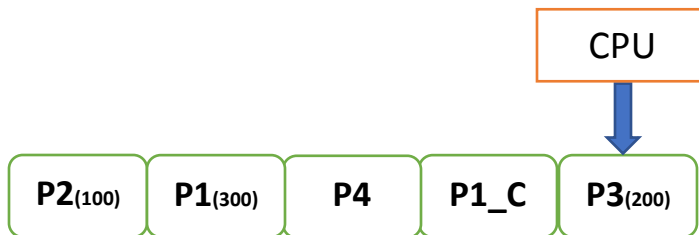
**T = 500**



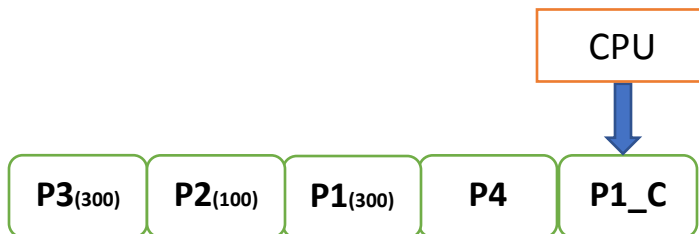
**T = 600**



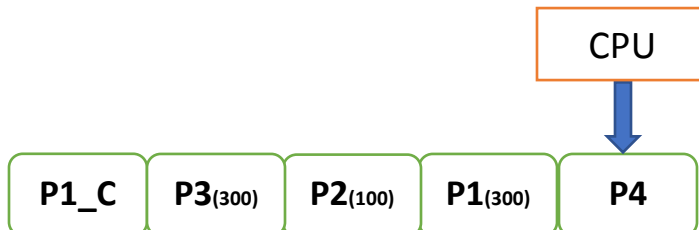
**T = 650**



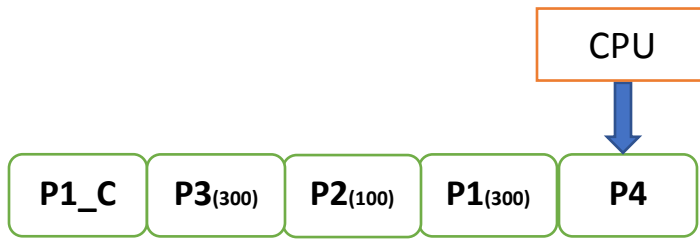
**T = 750**



**T = 900**



**T = 1000**



## سؤال ۵)

اگر به ترتیب از بالا به پایین و از سمت چپ به راست fork ها را شماره گذاری کنیم روند اجرای برنامه به صورت زیر خواهد بود:

با توجه به این شکل، عبارت شرطی اول که شامل سه fork است در نهایت منجر به ۴ بار اجرای fork4 می‌شود. یک بار اجرای fork4 باعث ۵ بار چاپ شدن عبارت hello می‌شود. پس در نهایت به اندازه ۴ ضربدر ۵ پردازش (۲۰ عدد) داریم که هر کدام در انتهای اجرایشان عبارت hello را چاپ میکنند.

