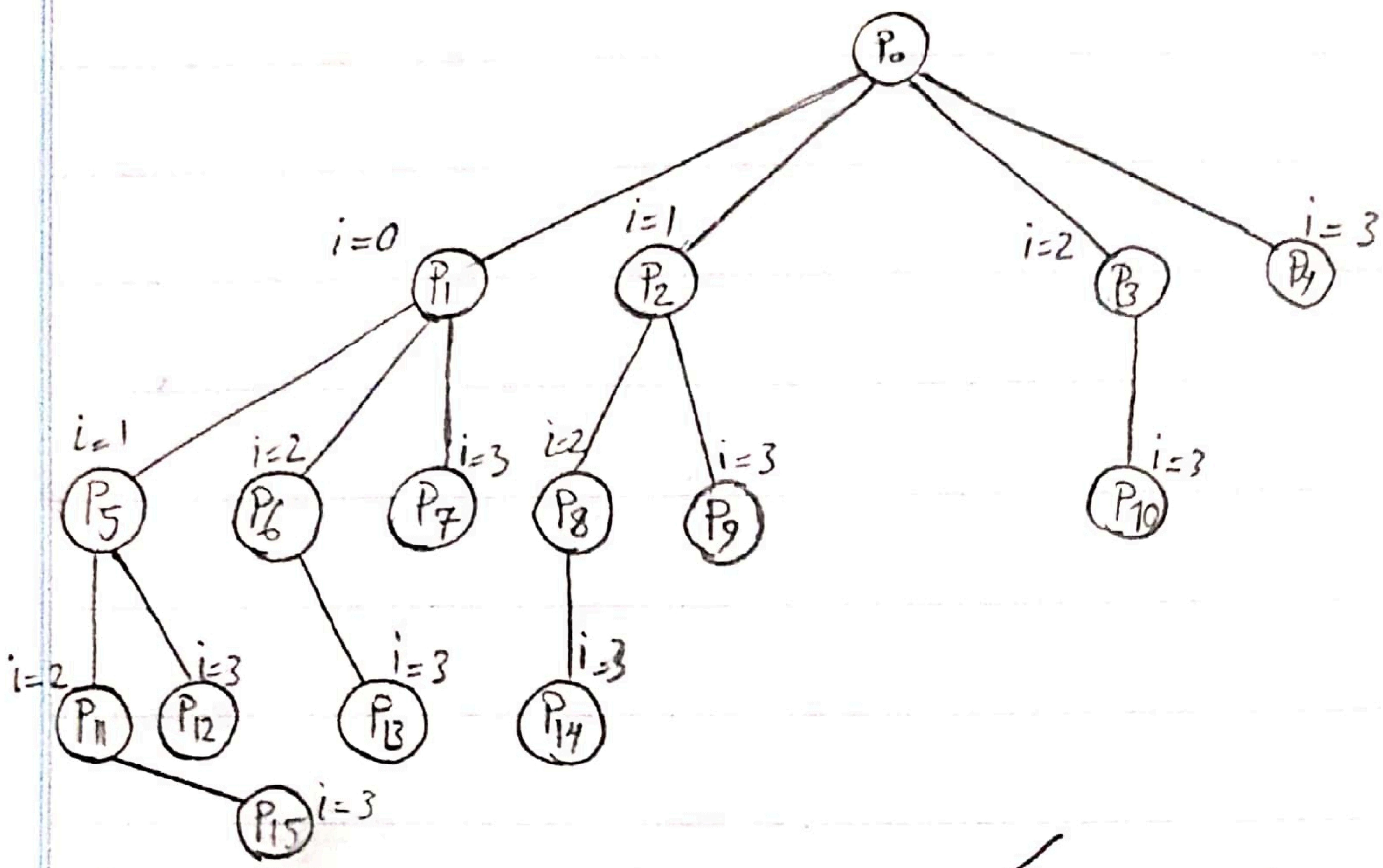
الناز نصاری 98411387

تمرین تئوری سری اول

(۱)



در اولین دور حلقه for ، یعنی زمانی که $i=0$ است، P_1 تولید می‌شود. وقتی $i=1$ می‌شود و دوباره به fork می‌رسیم، هرکدام از پراسس‌های P_0 و P_1 ، childهای P_2 و P_5 تولید می‌شوند. در دور بعدی ($i=2$)، هرکدام از پراسس‌های موجود (P_0 ، P_1 ، P_2 ، P_5)، childهای P_3 ، P_6 ، P_8 و P_{11} را تولید می‌کنند و در دور آخر، زمانی که به fork() رسیدیم، تمامی پراسس‌های موجود (P_0 ، P_1 ، P_2 ، P_5 ، P_3 ، P_6 ، P_8 ، P_{11})، childهای P_4 ، P_7 ، P_9 ، P_{10} ، P_{12} ، P_{13} ، P_{14} و P_{15} را تولید می‌کنند، یعنی در مجموع 15 پراسس تولید می‌شوند.

(2)

shared memory :

مزیت : shared memory ، سرعت بیشتری نسبت به message passing دارد، زیرا در shared memory ، واسطه ای برای ارسال و دریافت پیام‌ها وجود ندارد و وقت تلف نمی‌شود، اما در message pasing ، Karnel باید پیام‌ها را چک کند.

معایب : shared media ممکن است مشکلاتی از جمله همگام سازی و محافظت از حافظه را ایجاد کنند.

message passing :

مزایا: سخت افزار را به صورت موازی راحت تر می توان ساخت تا تاخیرهای ارتباطی بالاتر را کاملا تحمل کند و به علاوه پیاده سازی آن از shared media راحت تر است.

معایب : سرعت کمتری نسبت به shared media دارد. (به خاطر karnel)

برای داده های حجم بالا از shared memory و داده های حجم کم از message passing استفاده می کنیم. همچنین در distributed system از message passing و در سیستم های تک کامپیوتری از shared message استفاده می شود.

(3)

در multiprogramming ، OS اجازه می دهد بیشتر از یک برنامه همزمان روی یک CPU اجرا شوند. این کار باعث افزایش بهره وری CPU نیز می شود.

در OS multi tasking ، چندین process یا task همزمان با استفاده از چند CPU اجرا می شوند. در واقع CPU ، بین process ها ، switch می کند تا چند process یا task همزمان اجرا شوند.

(4) ، child processor مربوط به memory زیرا خواهد بوده 5 عدد ، کد خروجی parent مستقل از هم است.

5

آ) به سیگنالی که از دستگاه به CPU ارسال می‌شود interrupt می‌گوییم. هدف از interupt ، متوقف کردن CPU و انتقال execution به مکان مناسب هنگام دریافت سیگنال می‌باشد.

ب) trap ، سیگنالی است که توسط یک برنامه کاربر ارسال می‌شود به OS دستوری دهد تا با فاصله ، برخی از functionality ها را انجام دهد. اما interrupt ، سیگنالی به CPU است که توسط hardware ارسال می‌شود و event ها مشخص می‌کند که باید فوراً به آن توجه شود. در واقع ، trap نوعی interupt است که توسط نرم افزار تولید می‌شود، درصورتی که interrupt توسط سخت افزار ایجاد می‌شود.

ج) بله ، برنامه کاربر می‌تواند trap ایجاد کند. این کار به منظور یافتن خطاهای حسابی یا به بیان بهتر، فراخوانی روتین های OS انجام می‌شود. در واقع به CPU اعلام می‌کند که کاربر، سرویس ها درخواست کرده یا اتفاق غیرمنتظره در نرم افزار افتاده است.

6

Round robin برای سیستم های interactive بهتر است.

مورد الف ، الگوریتم round robin است و هر process زمان مشخص برای اجرای دستورات دارد و بنابراین ، این الگوریتم مناسب سیستم های interactive است. (سیستم های interactive سیستم هایی هستند با user سر و کار دارند و lag یا گیر کردن در آن ها خوشایند نیست.) یکی از معایب آن ، context switching overhead می باشد. (context switching

ذخیره state پردازه و جایگزینی آن با پردازه دیگر است.)

در مورد ، یکی از معایب آن ، این است که Starvation ممکن است اتفاق بیفتد.
برای مثال اگر یک پردازه داشته باشیم که زمان زیادی به CPU نیاز دارد در این
وسط پردازه های دیگری که زمان کمتری می گیرند وجود داشته باشند ، این پردازه ما ممکن است هیچ وقت
CPU نگیرد. overhead ، sort کردن آن هم داریم. (sort کردن و
sort نگه داشتن) . همچنین ، از مزایای این الگوریتم به سریع تر انجام شدن
فرآیندهای کوتاه اشاره کرد. بنابراین اگر فرآیندهای کوتاه داشته باشیم ،
روش SJF بهتری باشد.