**LAPORAN PRAKTIKUM**

**SISTEM OPERASI**

(**DOSEN PENGAMPU : IWAN LESMANA. S.KOM., M.KOM)**

**MODUL 6**



DISUSUN OLEH :

NAMA : MOHAMAD ABAN SY’BANA

NIM : 20230810012

KELAS : TINFC-2023-04

**TEKNIK INFORMATIKA**

**FAKULTAS ILMU KOMPUTER**

**UNIVERSITAS KUNINGAN**

**2024**

**Pretes**

1. Jelaskan apa yang dimaksud dengan penjadwalan proses dalam sistem operasi dan mengapa mekanisme ini penting ?

**Jawab :** Penjadwalan proses adalah metode yang digunakan oleh sistem operasi untuk menentukan proses mana yang akan dijalankan oleh CPU pada suatu waktu. Mekanisme ini penting karena membantu memaksimalkan penggunaan CPU, meningkatkan responsivitas aplikasi, memastikan distribusi sumber daya yang adil, dan mencegah kondisi deadlock.

1. Jelaskan perbedaan utama antara algoritma penjadwalan Preemptive Scheduling dan Non-preemptive Scheduling

**Jawab :**

* + Preemptive Scheduling memungkinkan sistem operasi untuk menghentikan sementara proses yang sedang berjalan dan memberikan CPU kepada proses lain yang memiliki prioritas lebih tinggi. Ini meningkatkan responsivitas dan efisiensi, tetapi bisa menimbulkan overhead karena konteks switching yang sering.
  + Non-preemptive Scheduling membiarkan proses yang sedang berjalan untuk menyelesaikan tugasnya sebelum memberikan CPU kepada proses lain. Ini lebih sederhana dan menghindari overhead dari switching, tetapi bisa menyebabkan proses tertentu harus menunggu lama (starvation).

1. Apa yang dimaksud dengan manajemen memori dalam sistem operasi? Sebutkan salah satu fungsi utama dari manajemen memori.

**Jawab :** Manajemen memori adalah proses mengelola penggunaan memori utama (RAM) oleh sistem operasi. Salah satu fungsi utama manajemen memori adalah alokasi memori, yang menentukan bagian memori mana yang akan digunakan oleh proses yang berbeda dan memastikan tidak ada proses yang saling mengganggu.

1. Berikan kelebihan dan kekurangan 3 metode alokasi memori

**Jawab :**

1. **Contiguous Allocation**:

* **Kelebihan**: Mudah diimplementasikan dan memiliki akses yang cepat.
* **Kekurangan**: Dapat menyebabkan fragmentasi eksternal dan membutuhkan relokasi memori jika tidak cukup ruang yang berdekatan.

1. **Paged Allocation**:

* **Kelebihan**: Mengatasi fragmentasi eksternal dan memori dapat dialokasikan secara tidak berurutan.
* **Kekurangan**: Overhead karena manajemen tabel halaman dan bisa menyebabkan fragmentasi internal.

1. **Segmented Allocation**:

* **Kelebihan**: Mendukung struktur data logis yang lebih baik dan mengurangi fragmentasi eksternal.
* **Kekurangan**: Kompleksitas dalam manajemen segmen dan potensi fragmentasi internal serta eksternal.

**Praktikum 1 penjadwalan proses**

1. Masuk pada compiler dan ketikan script kode berikut :

program LoopForeverTest

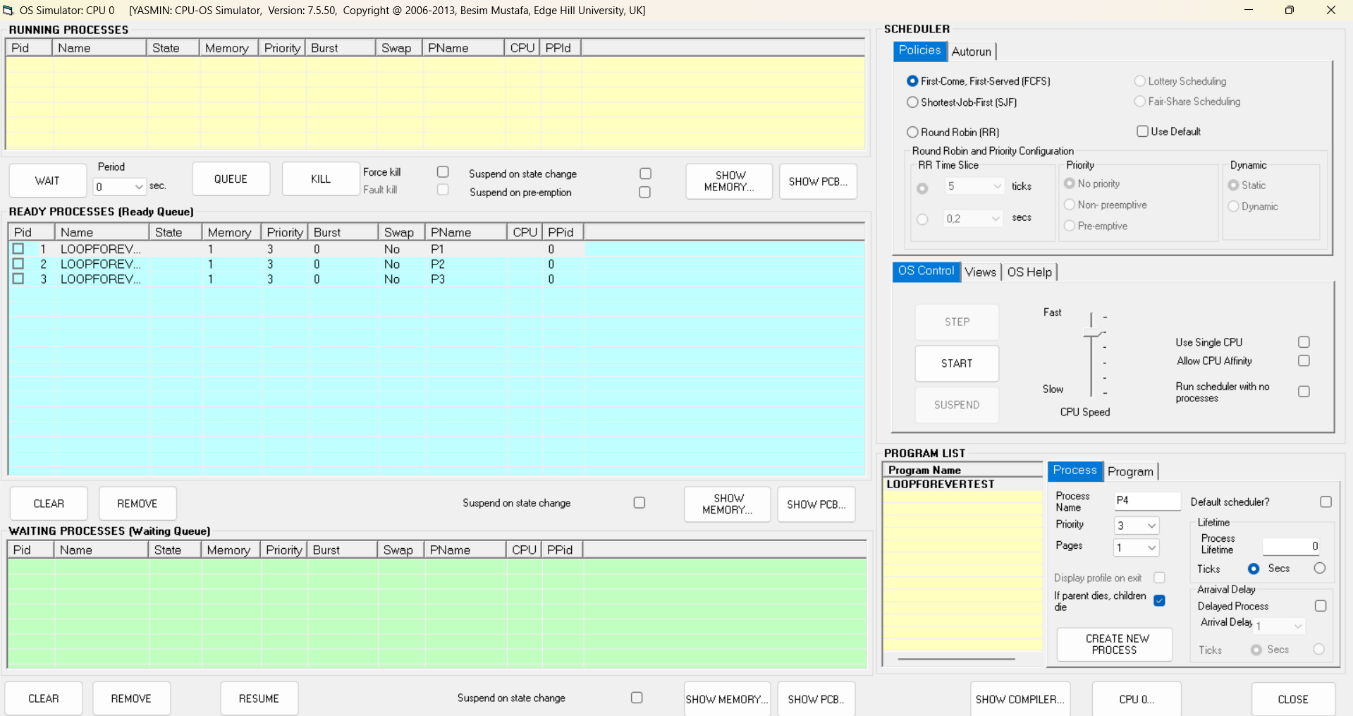
while true

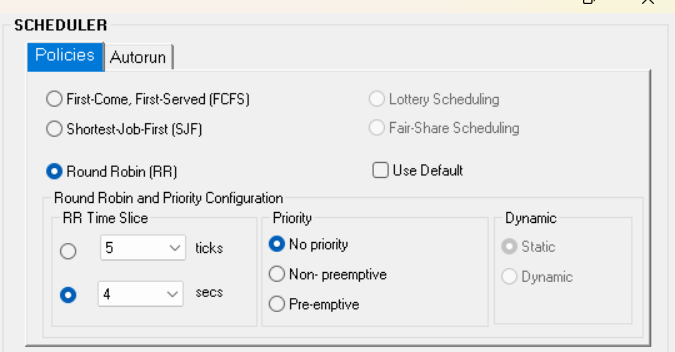
N = N + 1

wend

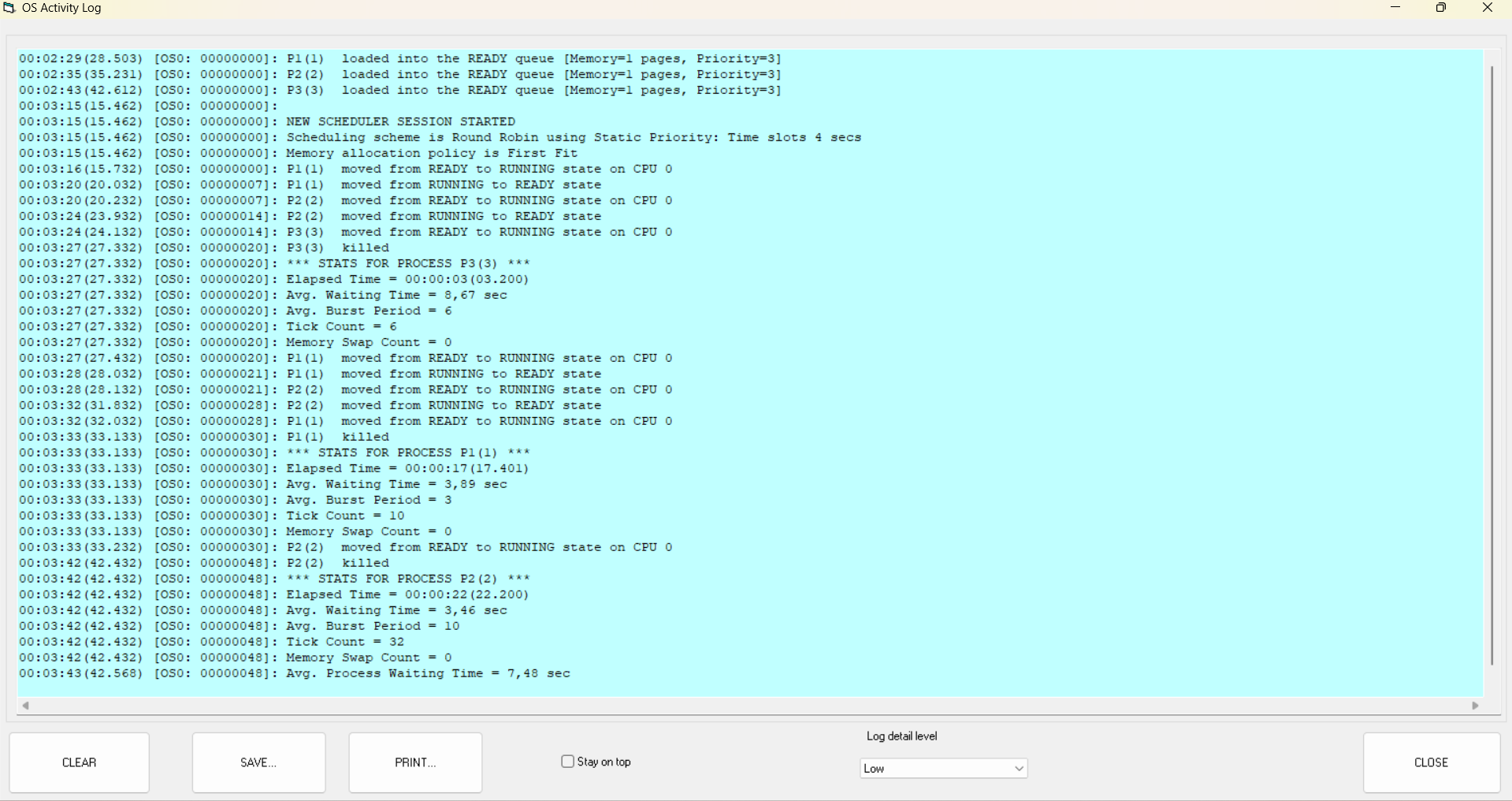
end

1. Beralihlah ke jendela OS Simulator.
2. Pilih program LoopForeverTest, selanjutnya kita akan membuat 3 proses, namun sebelum tiap proses dibuat aturlah nilai LIFETIME dan pilih SECS untuk tiap proses: 10 seconds, 32 seconds, 6 seconds.
3. Berikutnya memilih nilai timeslot melalui menu pull-down pada sub jendela SCHEDULER/POLICIES/RR Time Slice/Secs, pilih 4.





1. Tekan START dan tunggu hingga semua proses selesai.
2. Bukalah jendela OS ACTIVITY LOG dengan klik pada tombol VIEW LOG... Amati data log yang relevan kemudian perhatikan urutan pada proses yang berjalan (running processes) dan catat waktu yang dihabiskan oleh tiap proses selama dalam keadaan running



Pada log aktivitas OS ini, proses dijalankan menggunakan skema penjadwalan Round Robin dengan *time slice* sebesar 4 detik. Proses P1 memulai giliran pertama di CPU selama 4 detik (00:03:15.462 hingga 00:03:19.462), dilanjutkan oleh P2 (4 detik, 00:03:19.462 hingga 00:03:23.462), dan kemudian P3 (4 detik, 00:03:23.462 hingga 00:03:27.332). Setelah siklus pertama, P1 kembali ke status RUNNING selama dua giliran: pertama selama 4 detik (00:03:27.332 hingga 00:03:31.332), kemudian 2 detik sebelum selesai (00:03:31.332 hingga 00:03:33.133). Proses P2 juga menjalankan giliran kedua selama 4 detik (00:03:33.332 hingga 00:03:37.332), sementara P3 hanya menjalankan satu giliran sebelum di-*killed*.

Secara keseluruhan, total waktu yang dihabiskan untuk masing-masing proses dalam status RUNNING adalah: P1 selama 10 detik, P2 selama 8 detik, dan P3 hanya 4 detik. Penjadwalan berjalan efisien dengan mematuhi *time slice*, dan setiap proses mendapatkan alokasi waktu yang adil sebelum beralih ke status READY atau KILLED. Skema ini mencerminkan desain Round Robin yang mengutamakan distribusi CPU secara merata antar proses.

**Praktikum 2 pejadwalan proses**

1. Masuk pada compiler dan ketikan script kode berikut :

program LoopForeverTest2

while true

n = n + 1

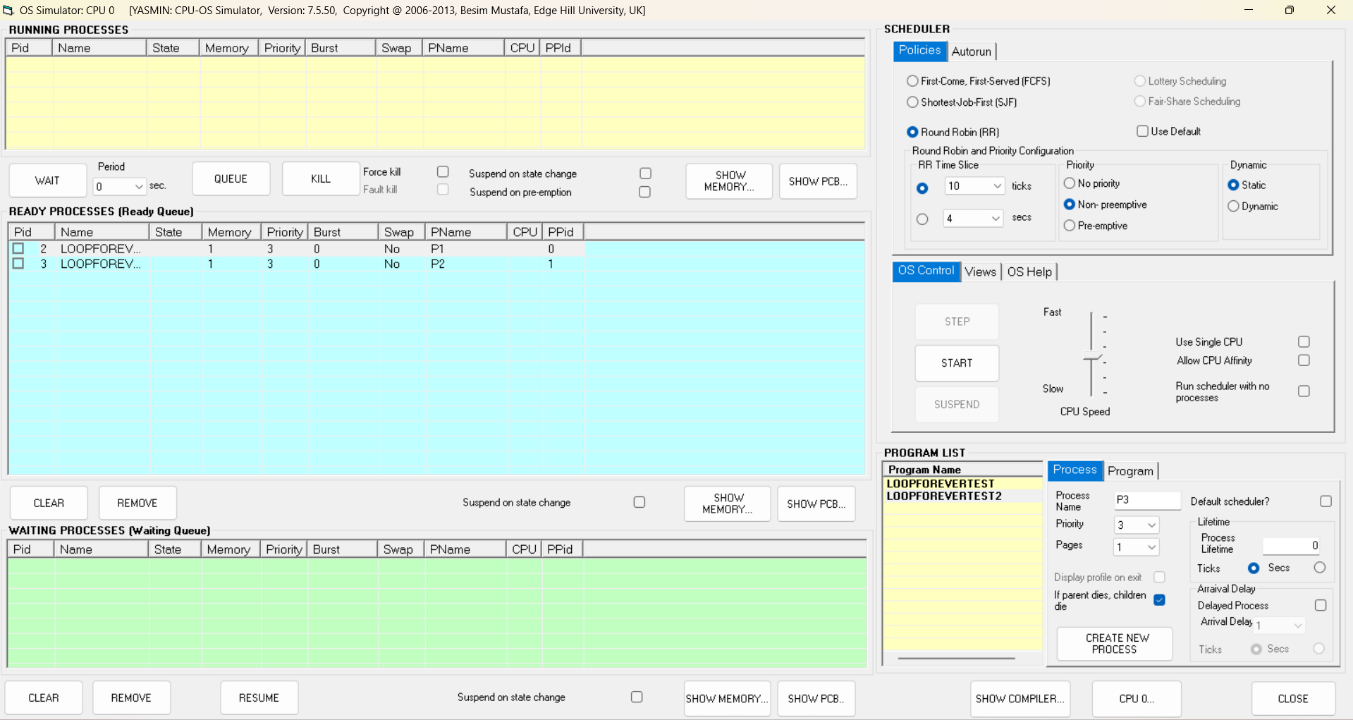
i = i + n

p = n + i + 5

wend

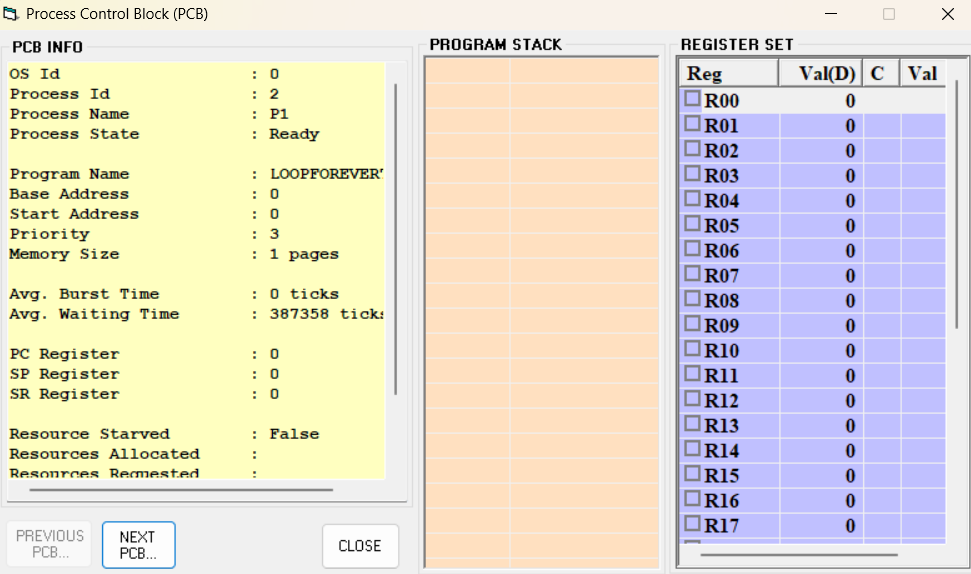
end

1. Compile kemudian muatkan ke memori kode diatas
2. Berailihlah ke OS Simulator.
3. Pada program list akan tampak 2 program: Loop ForeverTest, dan LoopForeverTest2.
4. Klik Loop ForeverTest dan buatlah satu proses.
5. Klik Loop ForeverTest2 dan buat satu proses.
6. Sekarang pada antrian ready seharusnya ada 2 proses
7. Pilih penjadwalan ROUND ROBIN, tipe prioritas NON-PREEMPTIVE dan RR Time Slice adalah 10 tick
8. atur pada setengah kecepatan simulasi.

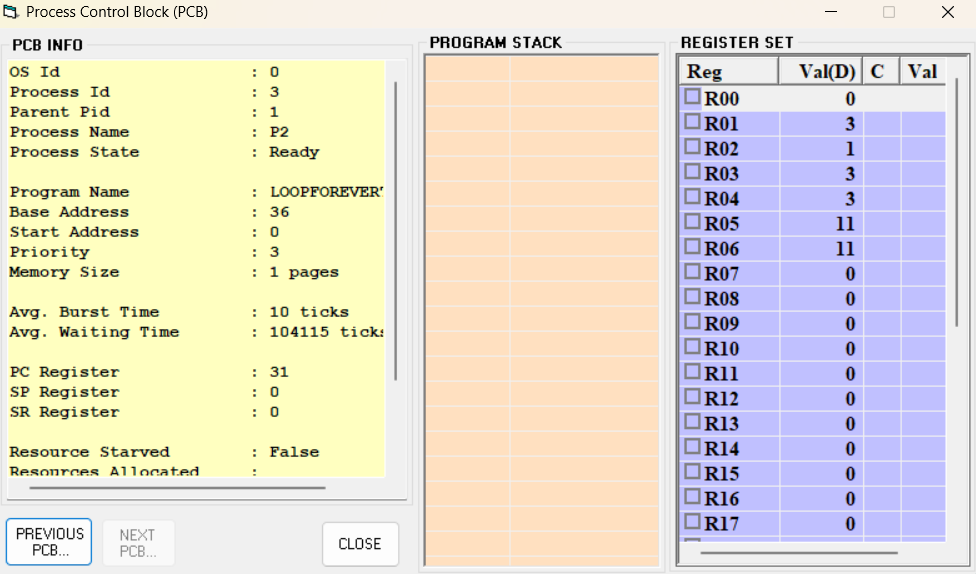


Lakukan langkah-langkah percobaan berikut ini:

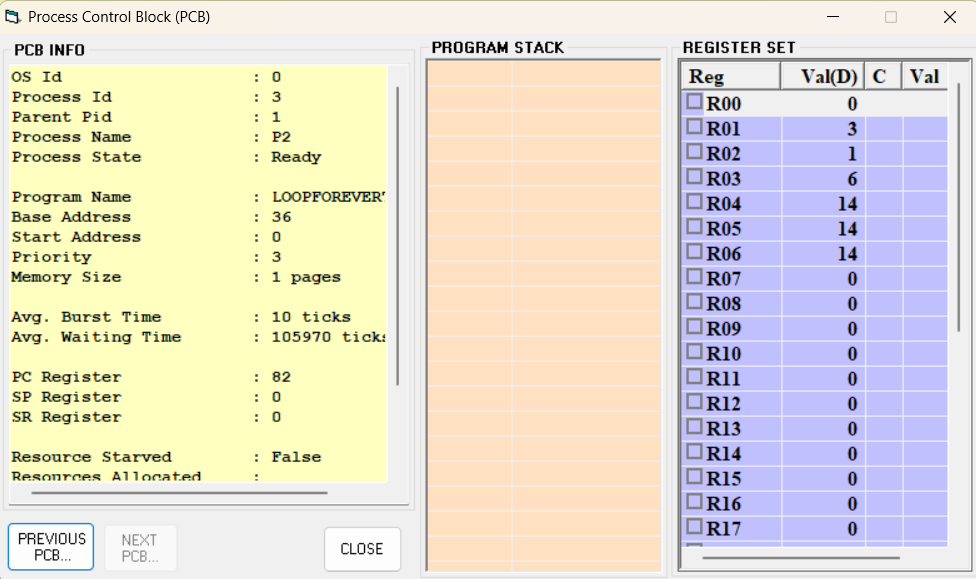
1. Sorot proses kemudian klik tombol PCB... amati nilai pada PC REGISTERS (pada PCB Info) pada masing-masing proses, dan catatlah.



1. Klik START dan siapkan mouse anda pada tombol SUSPEND (jangan klik).
2. Sewaktu running process kembali ke ready queue, segera klik tombol SUSPEND
3. Sorot proses pada antrian ready, dan klik tombol PCB... pada antrian ready. Catatlah nilai PC REGISTER nya.



1. Sekarang, beri check pada SUSPEND ON RUN dalam tampilan RUNNING kecepatan OS simulation. Klik tombol RESUME dan amati ketika proses yang antri kembali dalam keadaan running, yang menyebabkan simulasi berhenti (suspend) secara otomatis.
2. Perhatikan baik-baik pada kotak PC REGISTER dalam jendela CPU Simulator. Sekarang klik RESUME pada OS Simulator. Catatlah nilai pada PC REGISTER ketika nilainya berubah. Berikan pendapat anda terhadap nilai tersebut dan jelaskan apa yang terjadi.



Ketika PC Register berubah, ini menunjukkan bahwa CPU sedang melanjutkan eksekusi proses berdasarkan instruksi yang berada pada alamat memori tertentu. Jika perubahan ini terjadi secara berkala, itu mencerminkan eksekusi instruksi secara normal dalam siklus fetch-decode-execute. Jika nilai PC Register meningkat secara linear, berarti instruksi sedang dieksekusi secara berurutan. Namun, jika nilainya meloncat, ini mungkin mencerminkan operasi seperti percabangan (*branch*) atau pemanggilan prosedur (*function call*). Instruksi yang dieksekusi ini selaras dengan penjadwalan proses dan kondisi sistem pada waktu tersebut.

**Praktikum 3 manajemen memori**

1. Masukkan pada compiler dan ketikan script kode berikut :

program LoopForeverTest

while true

I = 1

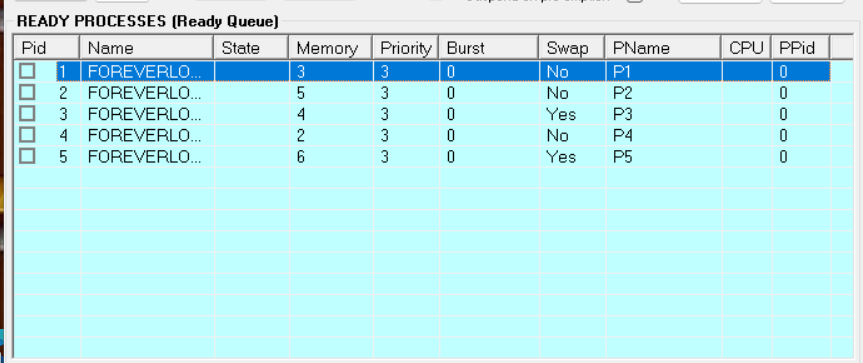
wend

end

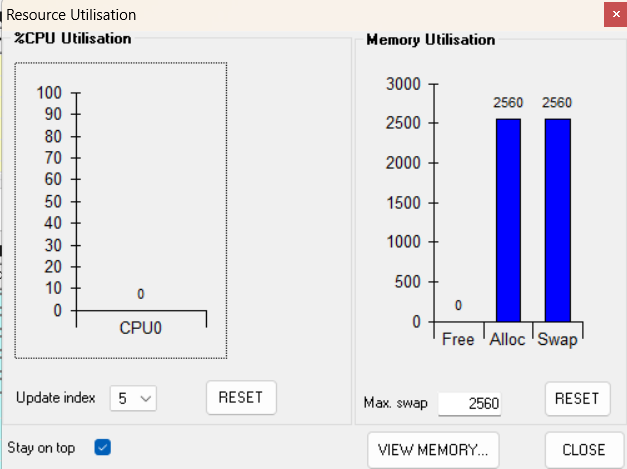
1. Klik tombol view memory aktifkan STAY ON TOP. Matikan PAGING ENABLED (dengan menghilangkan tanda check). Pilih program ForeverLoop, buatlah proses dengan ketentuan berikut:

|  |  |
| --- | --- |
| **Proses** | **Page** |
| P1 | 3 |
| P2 | 5 |
| P3 | 4 |
| P4 | 2 |
| P5 | 6 |

Perhatikan 4 proses yang terdapat pada jendela READY PROCESS. Amati data pada kolom SWAP. Beberapa tertulis "Yes", jelaskan artinya. Bandingkanlah dengan tampilan page yang tampil pada jendela MAIN MEMORY.



d. Klik VIEW UTILIZATION... informasi apa yang anda lihat? Klik STAY ON TOP. Sekarang, pada jendela OS Simulator, atur kecepatan pada kisaran 90, tekan START.



* + - 1. Jelaskan kejadian yang tampak pada jendela MAIN MEMORY (RAM).



**Jawab :** Jendela menunjukkan bahwa setiap proses dibagi menjadi halaman (256 bytes per halaman) dan ditempatkan di frame-frame pada RAM. Semua frame RAM telah terisi penuh oleh halaman dari berbagai proses (PID 1, PID 2, dan PID 4). Halaman proses yang tidak muat di RAM dipindahkan ke *swap memory*, seperti yang terlihat pada status *Allocated* dan *Swapped* sebesar 2560 bytes masing-masing.

1. Jelaskan kejadian yang tampak pada jendela MAIN MEMORY (RAM).Apa yang tampak pada kolom SWAP di jendela RUNNING PROCESSES? Mengapa selalu sama? Apakah ada kalanya berbeda dengan yang ditampilkan pada kolom SWAP di jendela READY PROCESSES? Jelaskan.

**Jawab :** Kolom SWAP di RUNNING PROCESSES menunjukkan status apakah ada halaman proses yang berada di *swap*. Status ini selalu mencerminkan data asli, dan saat proses berpindah dari READY ke RUNNING, halaman aktifnya diutamakan masuk ke RAM. READY dan RUNNING bisa memiliki perbedaan status *swap* tergantung pada alokasi memori saat itu.

1. Amati bagaimana proses-proses yang ready berganti (swap) keadaan dari waktu ke waktu. Beri penjelasannya.

**Jawab :** READY PROCESSES berganti status menjadi *swap* "Yes" jika halaman proses dipindahkan ke memori *swap* karena RAM penuh. Perubahan ini bergantung pada algoritma manajemen memori (seperti FIFO) yang mengganti halaman lama di RAM dengan halaman baru dari proses prioritas.

1. Informasi apa yang anda peroleh dari jendela RESOURCE UTILISATION terkait dengan manajemen memori?

**Jawab :** Jendela RESOURCE UTILISATION menunjukkan bahwa CPU tidak aktif (0% digunakan), sementara RAM penuh (2560 bytes dialokasikan) dan memanfaatkan *swap memory*. Hal ini menandakan teknik *paging* aktif dan memori utama sudah mencapai kapasitas maksimum.

**Postes**

* 1. Bagaimana pemilihan algoritma penjadwalan dapat mempengaruhi kinerja sistem operasi secara keseluruhan, terutama dalam konteks manajemen sumber daya CPU.

Jawab :

* + **Responsiveness**: Algoritma seperti Round-Robin memungkinkan sistem tetap responsif dengan memberikan setiap proses waktu eksekusi yang adil secara berurutan.
  + **Utilisasi CPU**: Algoritma seperti Shortest Job First (SJF) atau Priority Scheduling dapat memaksimalkan utilisasi CPU dengan menjalankan proses yang paling penting atau paling singkat terlebih dahulu.
  + **Throughput**: Algoritma yang efisien meningkatkan jumlah proses yang selesai dalam periode waktu tertentu, meningkatkan throughput sistem.
  + **Wait Time dan Turnaround Time**: Algoritma yang dipilih dapat meminimalkan waktu tunggu dan waktu penyelesaian untuk proses, seperti yang dilakukan oleh algoritma Multi-Level Queue Scheduling.
  + **Fairness**: Algoritma seperti Fair-Share Scheduling memastikan semua proses mendapat kesempatan untuk menggunakan CPU, mencegah starvation.
  1. Bagaimana penggunaan metode page swapping dan alokasi memori dapat mempengaruhi efisiensi penggunaan memori dalam sistem operasi?

**Jawab :**

* **Page Swapping**:
* **Pengertian**: Mengacu pada proses memindahkan halaman memori yang tidak aktif ke penyimpanan sekunder untuk membuat ruang bagi halaman baru yang dibutuhkan.
* **Efisiensi**: Memungkinkan sistem menggunakan memori fisik dengan lebih baik, menghindari kelebihan beban memori, namun bisa menyebabkan overhead jika swapping terlalu sering terjadi.
* **Thrashing**: Terlalu banyak swapping dapat menyebabkan penurunan kinerja drastis karena waktu lebih banyak dihabiskan untuk memindahkan data daripada mengeksekusi proses.
* **Alokasi Memori**:
* **Contiguous Allocation**: Alokasi dalam blok yang berdekatan; sederhana namun rentan terhadap fragmentasi eksternal.
* **Paged Allocation**: Memecah memori ke dalam halaman kecil; mengatasi fragmentasi eksternal namun bisa menyebabkan fragmentasi internal.
* **Segmented Allocation**: Memecah memori ke dalam segmen berdasarkan fungsi program; memudahkan manajemen memori untuk program modular, namun lebih kompleks untuk diimplementasikan.

**Tugas**

1. Buatlah penjadwalan proses dengan membuat 3 proses, namun sebelum tiap proses dibuat aturlah nilai LIFETIME dan pilih SECS untuk tiap proses: 15 seconds, 37 seconds, 11 seconds. Dengan langkah-langkah sesuai dengan contoh diatas!
2. Masuk pada compiler dan ketikan script kode berikut :

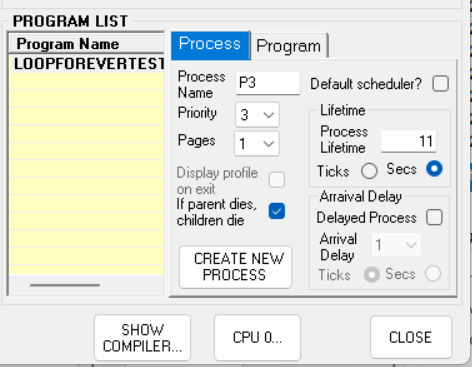
program LoopForeverTest

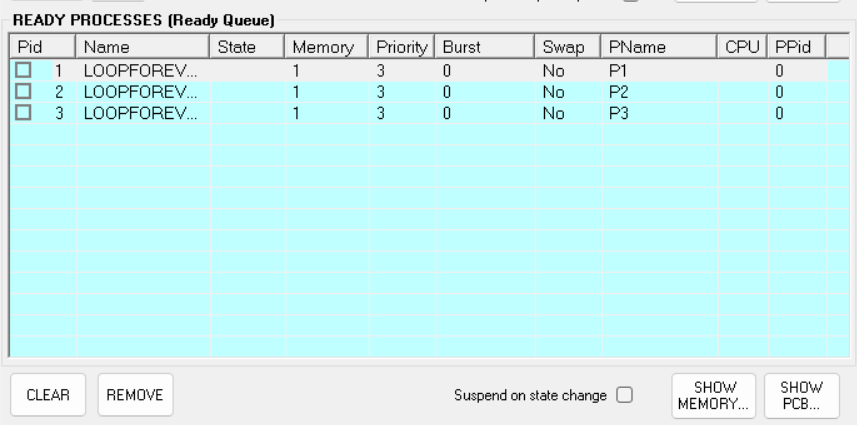
while true

N = N + 1

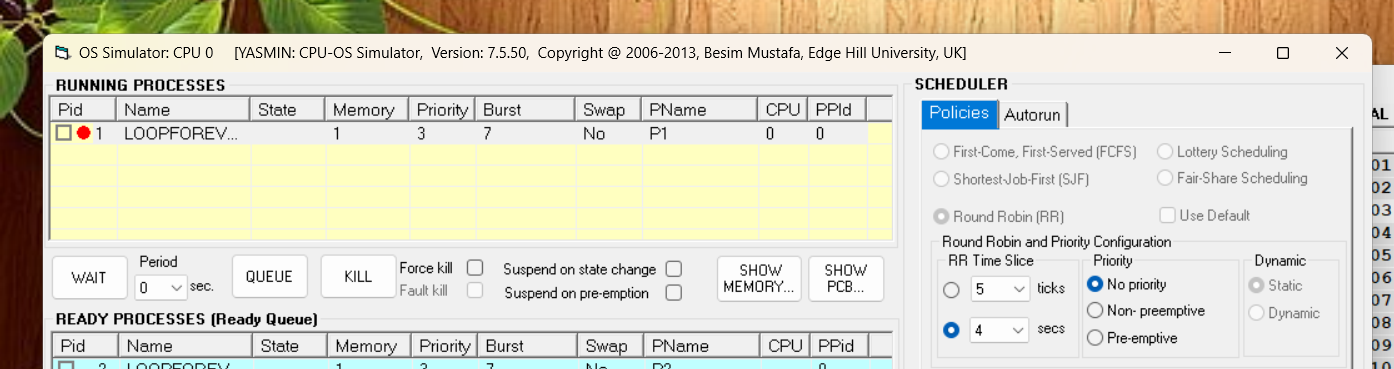
wend

end

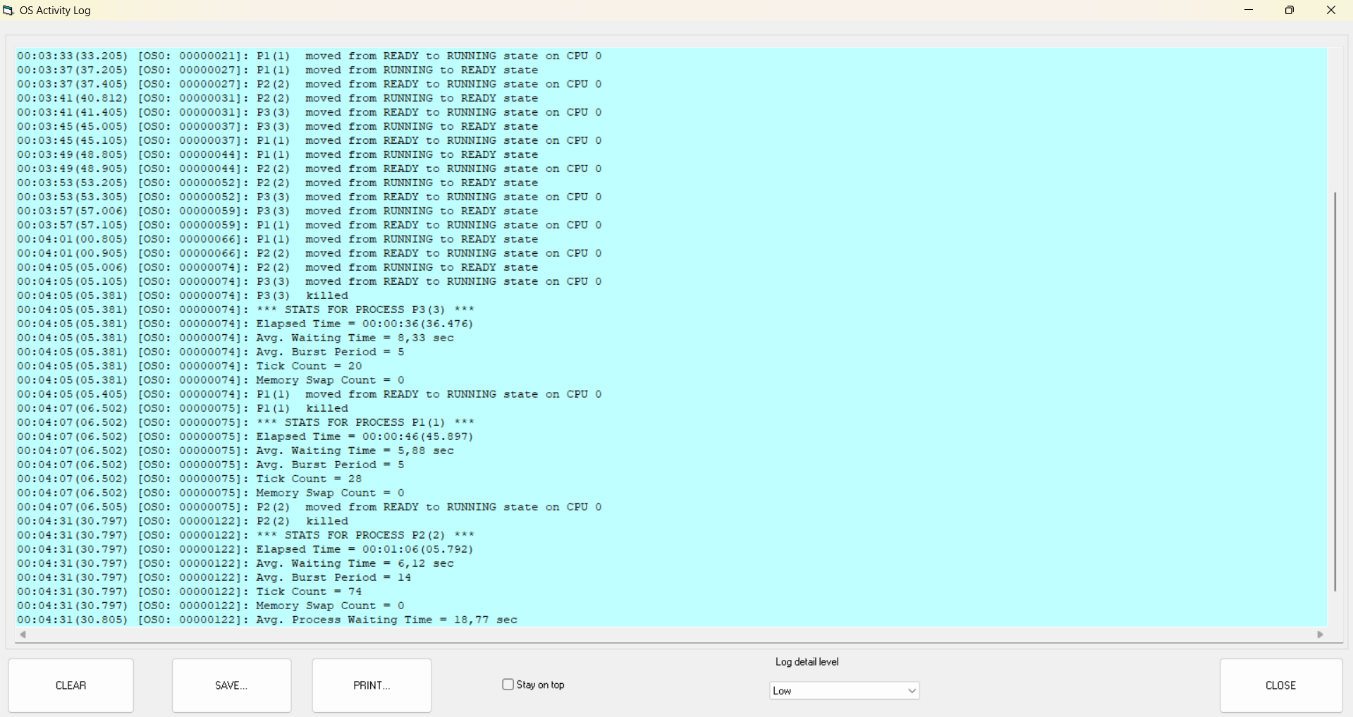
1. Beralihlah ke jendela OS Simulator.
2. Pilih program LoopForeverTest, selanjutnya kita akan membuat 3 proses, namun sebelum tiap proses dibuat aturlah nilai LIFETIME dan pilih SECS untuk tiap proses: 15 seconds, 37 seconds, 11 seconds. 



1. Berikutnya memilih nilai timeslot melalui menu pull-down pada sub jendela SCHEDULER/POLICIES/RR Time Slice/Secs, pilih 4.



1. Tekan START dan tunggu hingga semua proses selesai.
2. Bukalah jendela OS ACTIVITY LOG dengan klik pada tombol VIEW LOG... Amati data log yang relevan kemudian perhatikan urutan pada proses yang berjalan (running processes) dan catat waktu yang dihabiskan oleh tiap proses selama dalam keadaan running.



Log ini mencatat aktivitas beberapa proses yang bergantian menjalankan tugas pada CPU. Proses P1(1) pertama kali berjalan selama 4 detik, kemudian diikuti oleh P2(2) dan P3(3), masing-masing berjalan sekitar 4 detik. Setelah siklus pertama selesai, proses kembali mendapatkan waktu berjalan dengan durasi yang bervariasi. P1(1) berjalan lagi selama 4,3 detik, sementara P2(2) berjalan 2,4 detik sebelum aktivitasnya selesai. Proses P3(3) tidak dilanjutkan setelah 00:04:05 dan dinyatakan "killed", begitu pula dengan P1(1), yang berakhir pada 00:04:07.

Secara keseluruhan, total waktu CPU yang dihabiskan oleh setiap proses adalah 8,3 detik untuk P1(1), 6,4 detik untuk P2(2), dan 4,7 detik untuk P3(3). Log ini menunjukkan bagaimana CPU mengelola proses menggunakan mekanisme penjadwalan untuk berbagi waktu, serta menangani proses yang tidak lagi aktif atau dihentikan.

**Kesimpulan**

Penjadwalan Proses dan Manajemen Memori adalah dua konsep penting dalam sistem operasi yang saling berkaitan dan berperan besar dalam efisiensi dan performa sistem.

Penjadwalan Proses adalah metode yang digunakan sistem operasi untuk menentukan proses mana yang harus dijalankan oleh CPU pada suatu waktu tertentu. Algoritma penjadwalan seperti *preemptive* dan *non-preemptive* memainkan peran penting dalam hal ini. *Preemptive Scheduling* memungkinkan sistem operasi untuk menghentikan sementara proses yang sedang berjalan dan memberikan CPU kepada proses lain yang lebih penting atau memiliki prioritas lebih tinggi, meningkatkan responsivitas dan efisiensi. Sebaliknya, *Non-preemptive Scheduling* membiarkan proses yang sedang berjalan menyelesaikan tugasnya sebelum memberikan CPU kepada proses lain, menghindari overhead dari switching tetapi bisa menyebabkan proses tertentu harus menunggu lama (starvation).

Manajemen Memori adalah proses mengelola penggunaan memori utama (RAM) oleh sistem operasi. Salah satu fungsi utama manajemen memori adalah alokasi memori, yang menentukan bagian memori mana yang akan digunakan oleh proses yang berbeda dan memastikan tidak ada proses yang saling mengganggu. Teknik seperti page swapping dan metode alokasi memori (contiguous, paged, segmented) sangat mempengaruhi efisiensi penggunaan memori. Page Swapping memungkinkan sistem menggunakan memori fisik dengan lebih baik, meskipun bisa menyebabkan overhead jika swapping terlalu sering terjadi.

**Hubungan Kedua Konsep**

Penjadwalan proses dan manajemen memori bekerja bersama untuk memastikan bahwa sistem operasi dapat mengelola sumber daya dengan efisien, mendukung kinerja optimal, dan memberikan pengalaman pengguna yang lebih baik. Pemilihan algoritma penjadwalan yang tepat dan metode alokasi memori yang efisien sangat penting untuk mencapai tujuan ini, memastikan proses berjalan lancar, CPU digunakan secara optimal, dan memori dikelola dengan baik.