LAPORAN PRAKTIKUM

SISTEM OPERASI

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

MODUL 2

DISUSUN OLEH :

NAMA : MOHAMAD ABAN SY'BANA

NIM : 20230810012

KELAS : TINFC-2023-04

TEKNIK INFORMATIKA

FAKULTAS ILMU KOMPUTER

UNIVERSITAS KUNINGAN

2024

PRE TEST

1. **Apa itu thread dalam konteks sistem operasi?**
   Jawab : Dalam konteks sistem operasi, thread adalah unit dasar penggunaan CPU yang terdiri dari beberapa komponen, seperti program counter, thread_ID, register set, dan stack. Thread merupakan unit pemrosesan terkecil pada CPU yang bersifat independen.

2. **Sebutkan dua karakteristik utama dari sebuah proses?**
   Jawab :
   - Memiliki ruang alamat sendiri yang terpisah dari proses lain, yang berarti setiap proses berjalan dalam lingkungan terisolasi.
   - Memiliki sumber daya yang dialokasikan khusus oleh sistem operasi, seperti memori, waktu CPU, dan akses ke perangkat input/output.

3. **Apa perbedaan antara proses dan thread?**
   Jawab : Proses adalah Program atau aplikasi yang berjalan secara independen di ruang alamatnya sendiri. Proses memiliki ruang memori dan sumber daya sendiri. Sistem operasi mengelola proses dan mengalokasikan sumber daya ke proses tersebut.sedangkan Thread adalah Unit eksekusi dalam suatu proses yang berjalan di ruang alamatnya sendiri. Thread adalah unit dasar yang berbagi memori yang sama dengan proses. Thread dapat menjalankan bagian mana pun dari kode proses, termasuk bagian yang sedang dijalankan oleh thread lain.

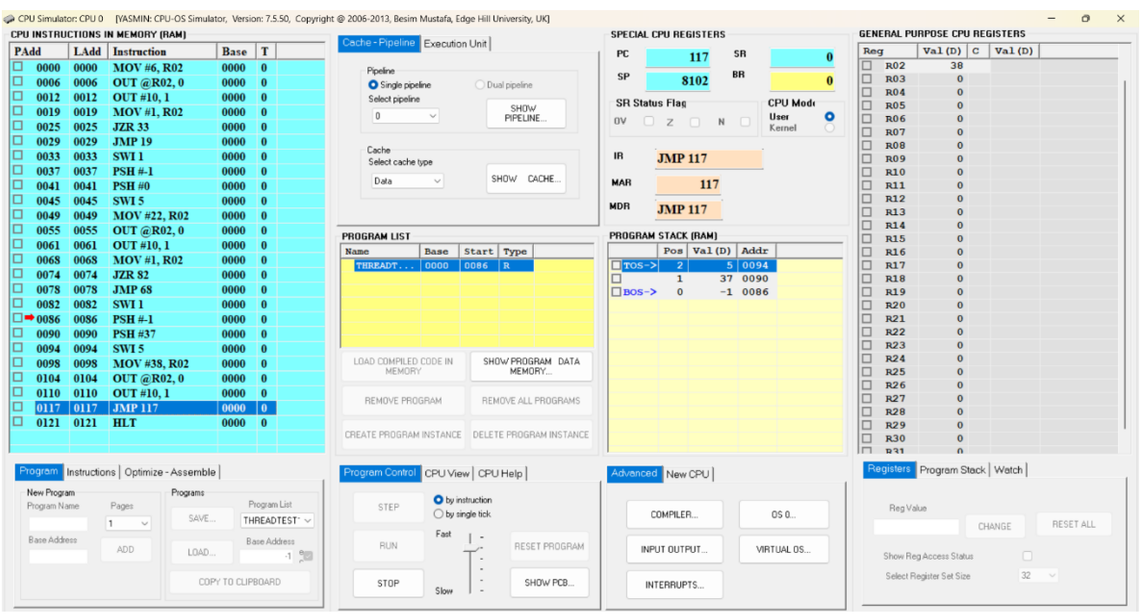
4. **Jelaskan fungsi dari program counter dalam sebuah thread ?**
   Jawab :
   Program counter (PC) dalam sebuah thread berfungsi untuk menandai urutan instruksi yang akan dijalankan:
   - PC adalah register khusus dalam CPU yang menyimpan alamat memori instruksi berikutnya yang akan dieksekusi.
   - Saat CPU mengambil instruksi, PC akan diperbarui dengan alamat memori instruksi berikutnya.
   - CPU kemudian mengambil instruksi tersebut dan melanjutkan eksekusi program secara berurutan.

PRAKTIKUM

1. Jalankan simulator. Tuliskan kode berikut:

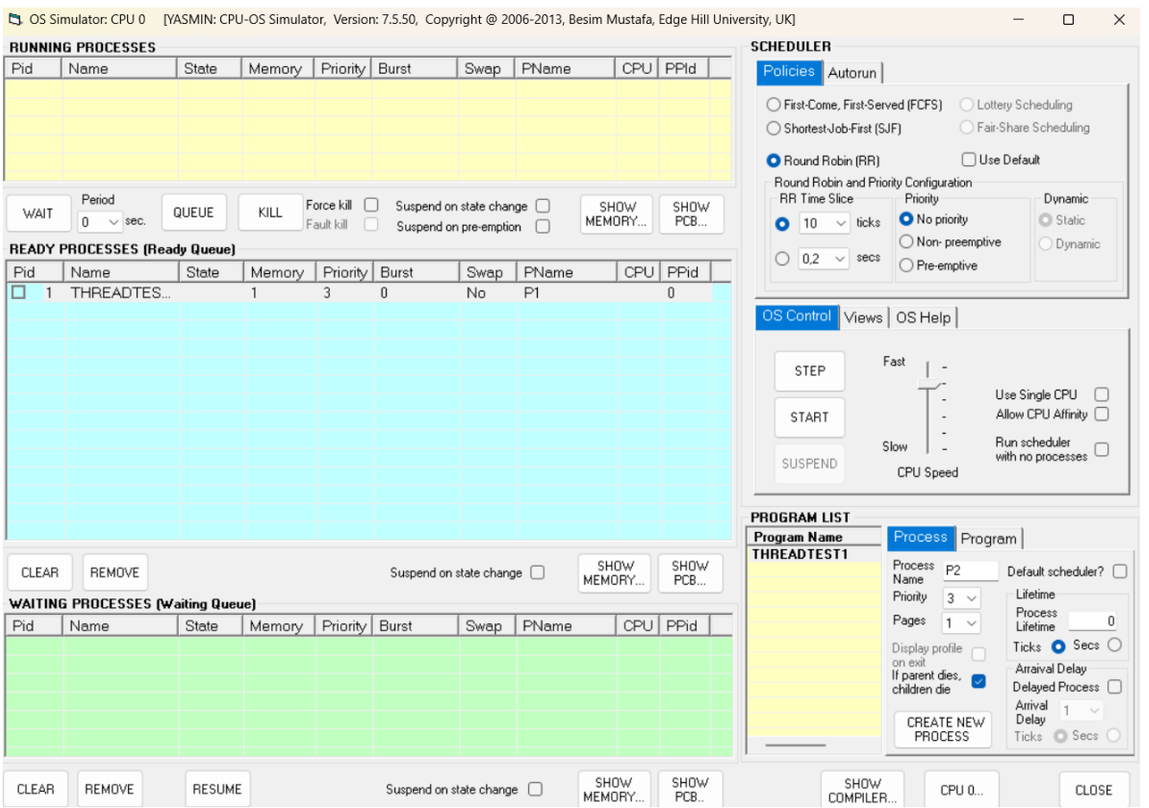
```
program ThreadTest1
sub thread1 as thread
        writeln("In thread1")
        while true
        wend
end sub
sub thread2 as thread
        call thread1
        writeln("In thread2")
        while true
        wend
end sub
call thread2
writeln("In main")
do
loop
end
```

1. Compile dan run kode program
   a. Compile dengan menekan tombol compile pada compile



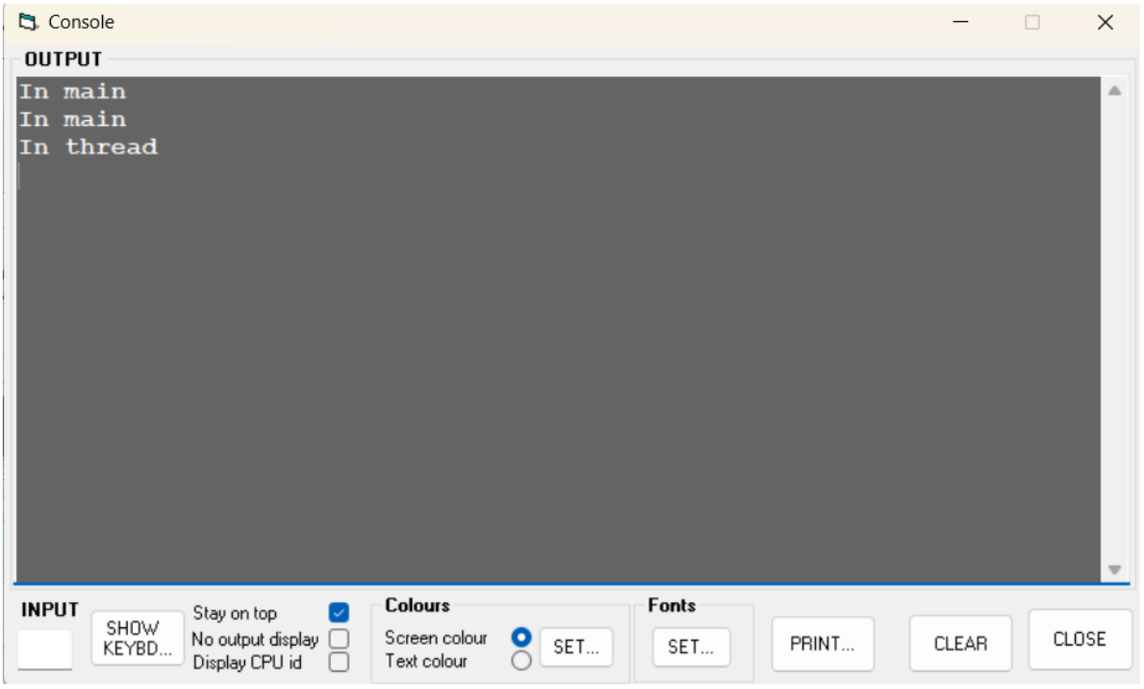
b. Tampilkan jendela konsol dengan klik tombol input/output dan aktifkan stay on top

c. Beralih ke os simulator, buat satu proses baru dengan create new proses.
d. Pastikan jenis penjadwalan adalah ROUND ROBIN, 10 ticks dengan kecepatan simulasi maksimum.

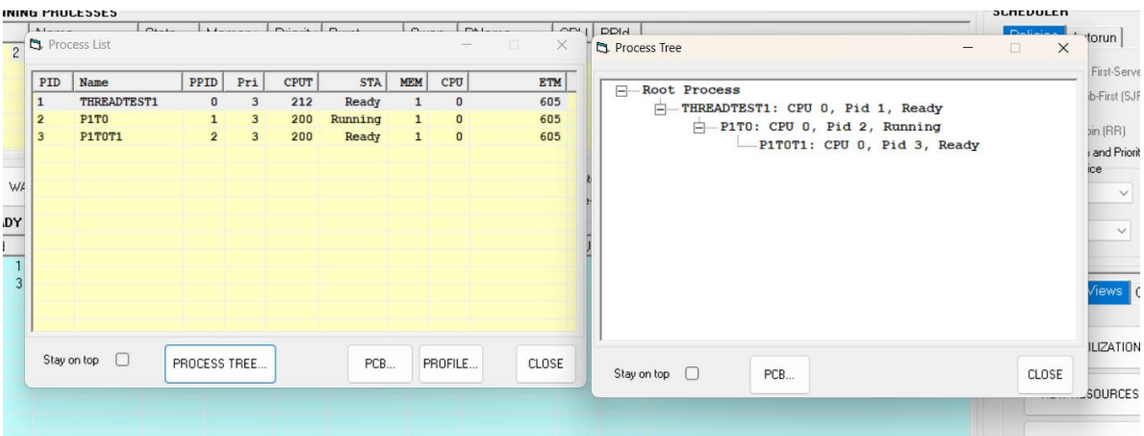


2. Tekan tombol start dan amatilah tampilan pada jendela console.
   a. Berapakah jumlah proses yang telah dibuat?
      Jawab : Jumlah proses yang di buat adalah 1.

   b. Sebutkan mana yang disebut proses dan mana yang disebut thread!
      Jawab : THREADTEST1 adalah proses utama dan P1T0 dan P1T0T1 adalah thread.



3. Klik tombol VIEW PROCESS LIST. Kemudian klik PROCESS TREE identifikasi proses induk dan proses turunan.



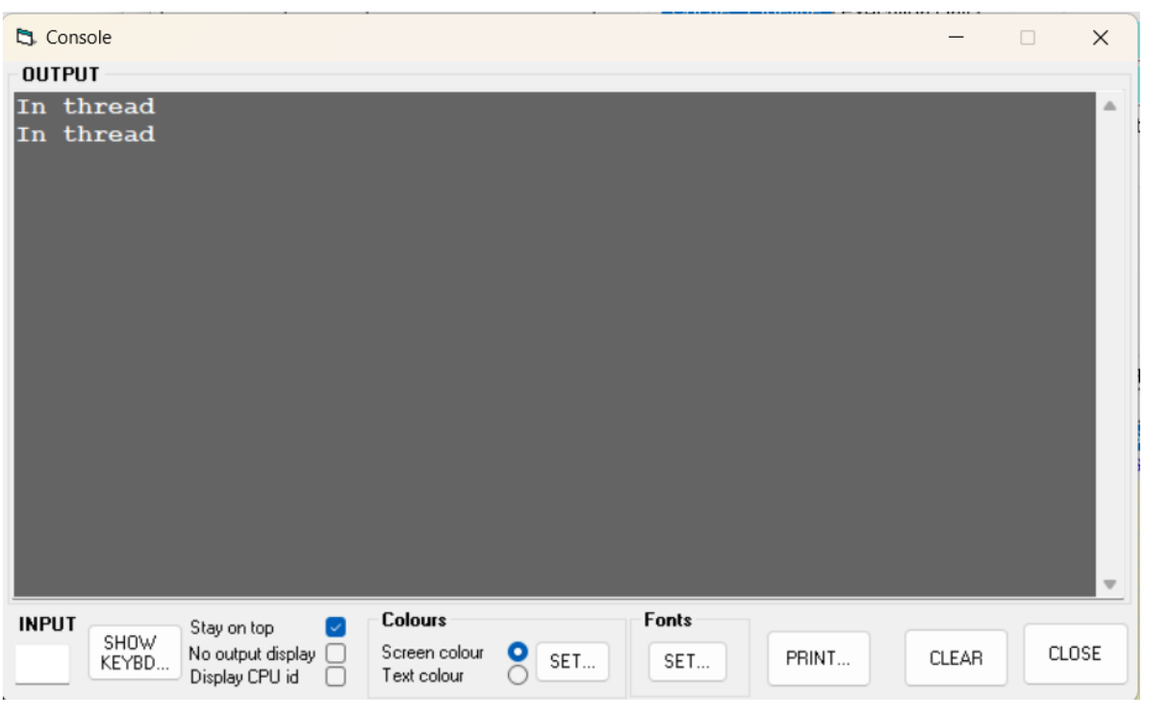
4. Gunakan tombol kill untuk menghentikan proses.

Memodifikasi kode sumber untuk membuat versi tanpa thread dan membandingkannya dengan versi yang menggunakan thread

1. Modifikasi source code ThreadTest1 dengan menghapus instance "as thread" pada deklarasi subroutine. Ganti nama program menjadi ThreadTest3.

```
program ThreadTes3
    sub thread1
        writeln("In thread")
        while true
    wend
    end sub
    sub thread2
        writeln("In thread")
        while true
        wend
    end sub
    call thread2
    writeln("In main")
    do
    loop
end
```

2. Compile code hasil modifikasi tersebut.
3. Buat proses baru lagi, kemudian jalankan di OS simulator.
4. Amati tampilan pada jendela console, apa perbedaan proses tanpa thread dan proses menggunakan thread?



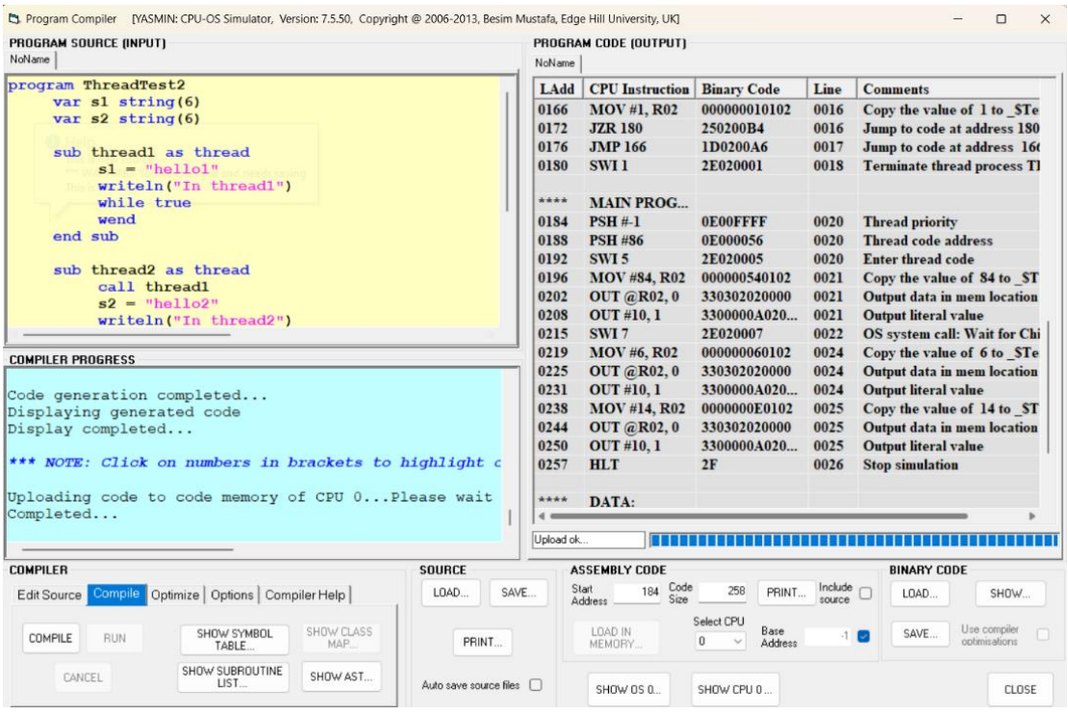
- Output di atas terdapat dua kali "In thread," yang menunjukkan bahwa thread telah dibuat dan dieksekusi beberapa kali.
- Proses Tanpa Thread **:** Program dieksekusi secara berurutan, satu instruksi pada satu waktu.
- Proses Menggunakan Thread : Program dapat mengeksekusi banyak instruksi secara bersamaan.

5. Gunakan tombol kill untuk menghentikan proses.

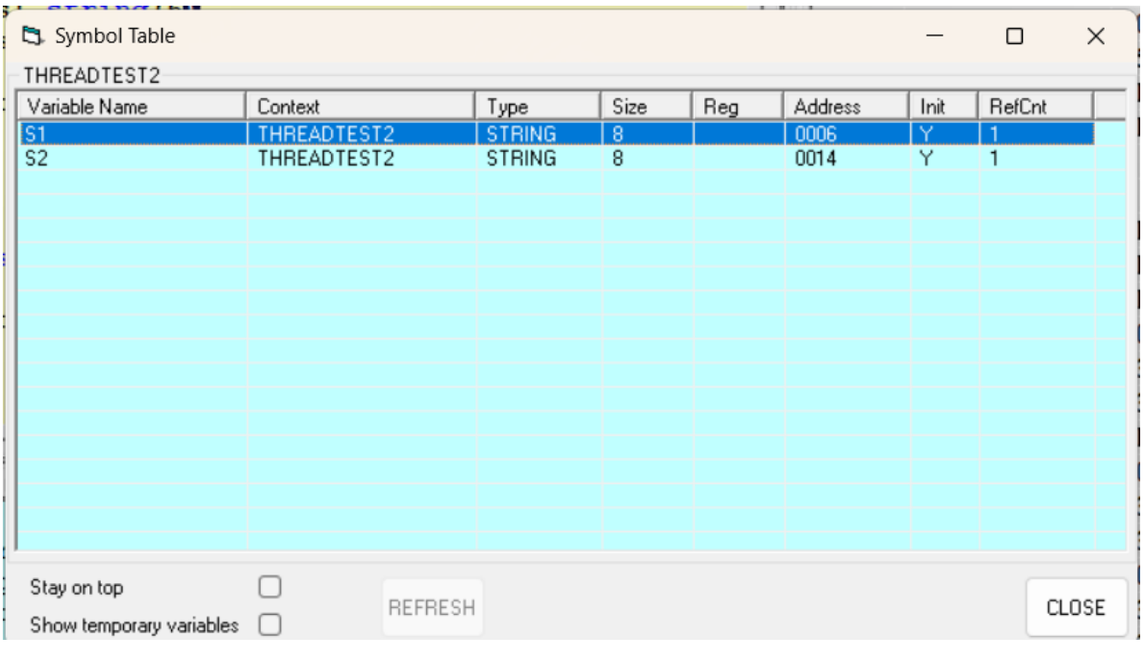
Memahami bagaimana thread-thread berbagi sumber daya dari proses induk.

1. Buatlah program baru dengan memodifikasi source code threadtest1 seperti sebgai berikut

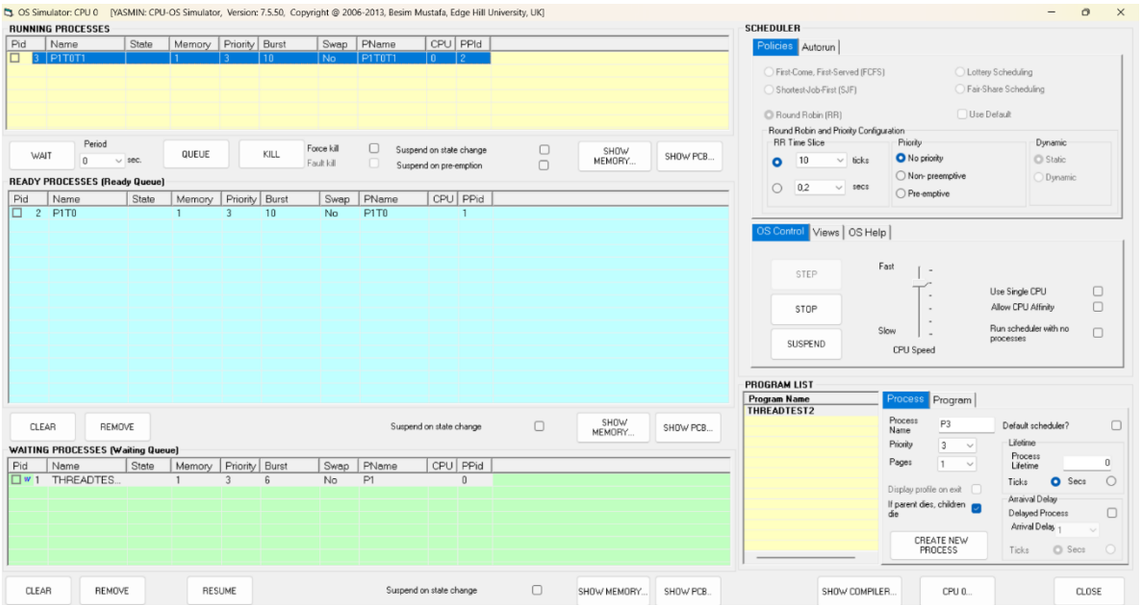
```
program ThreadTest2
    var s1 string (6)
    var s2 string (6)
    sub thread1 as thread
        s1 = "hello1"
        writeln("In thread1")
        while true
        wend
    end sub
    sub thread2 as thread
        call thread1
        s2 = "hello2"
        writeln("In thread2")
        while true
        wend
    end sub
    call thread2
    writeln("In main")
    wait
    writeln(s1)
    writeln(s2)
end
```



2. Compile program tersebut. Klik tombol SYMBOL TABLE pada jendela compiler. Amati tentang variable s1 dan s2.



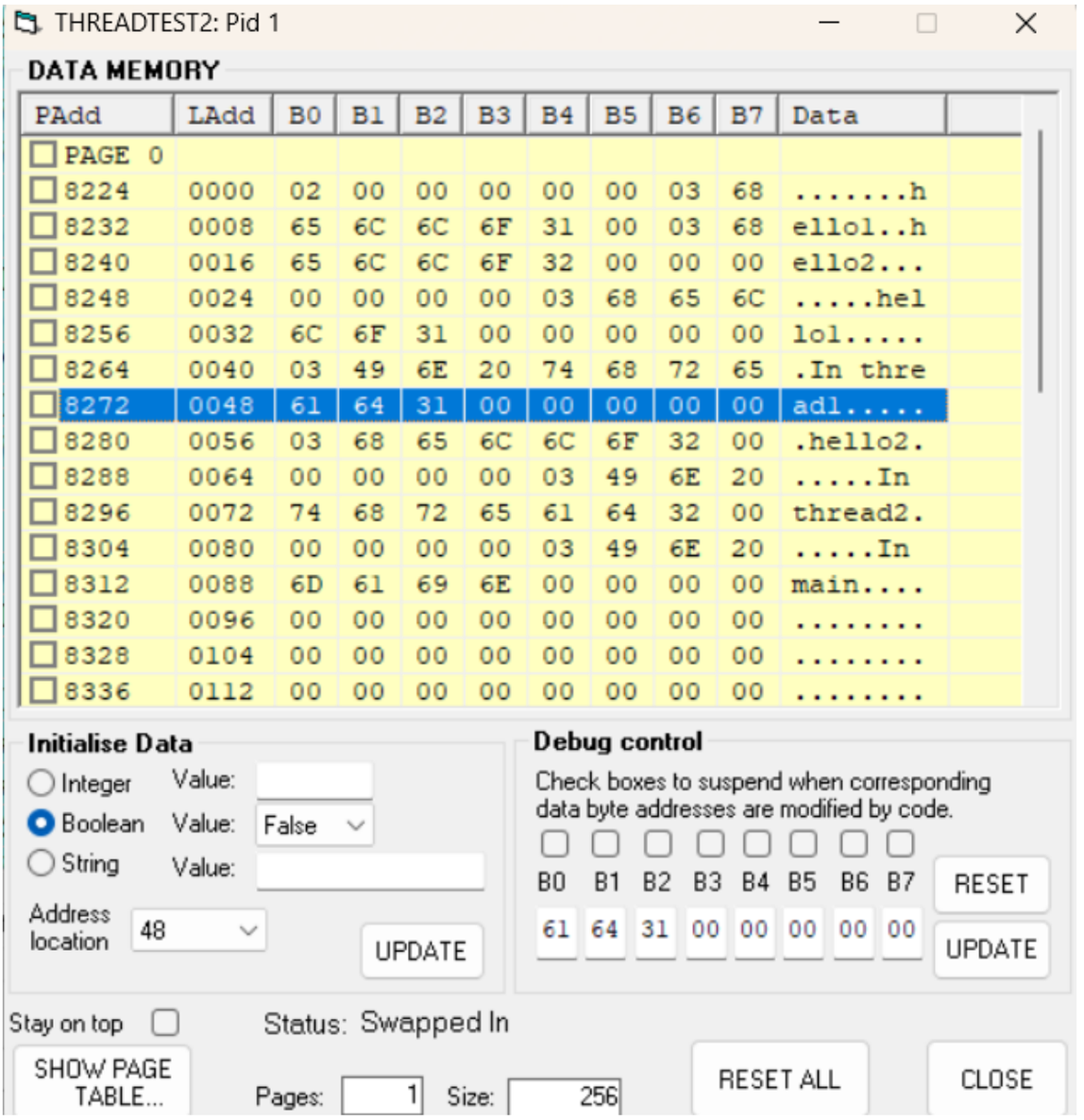
3. Tampilkan jendela konsol dengan klik tombol input/output dan aktifkan stay on top
4. Beralih ke os simulator, buat satu proses baru dengan create new proses



5. Pastikan jenis penjadwalan adalah ROUND ROBIN, 10 ticks dengan kecepatan simulasi maksimum

6. Tekan tombol start dan amatilah tampilan pada jendela console.

7. Tekan tombol show memory, amati alamat memory yang digunakan variable s1 dan s2. Simpulkan apa yang terjadi.



Gambar di atas Variabel s1 menyimpan string "hello1" dengan null byte (\0) di akhir string untuk menandai akhir dari teks. Variabel s2 menyimpan string "hello2" dengan null byte di awal dan di akhir string untuk menandai akhir dari teks. Ini menunjukkan bahwa string yang disimpan dalam memori menggunakan terminasi null (\0), yang umum digunakan dalam pemrograman untuk menandai akhir string. Hal ini juga memperlihatkan bahwa string disimpan berdekatan satu sama lain dalam memori, dengan masing-masing string memiliki rentang alamat memori yang unik.

POSTTES

1. **Jelaskan bagaimana thread-thread berbagi sumber daya dari proses induk.**

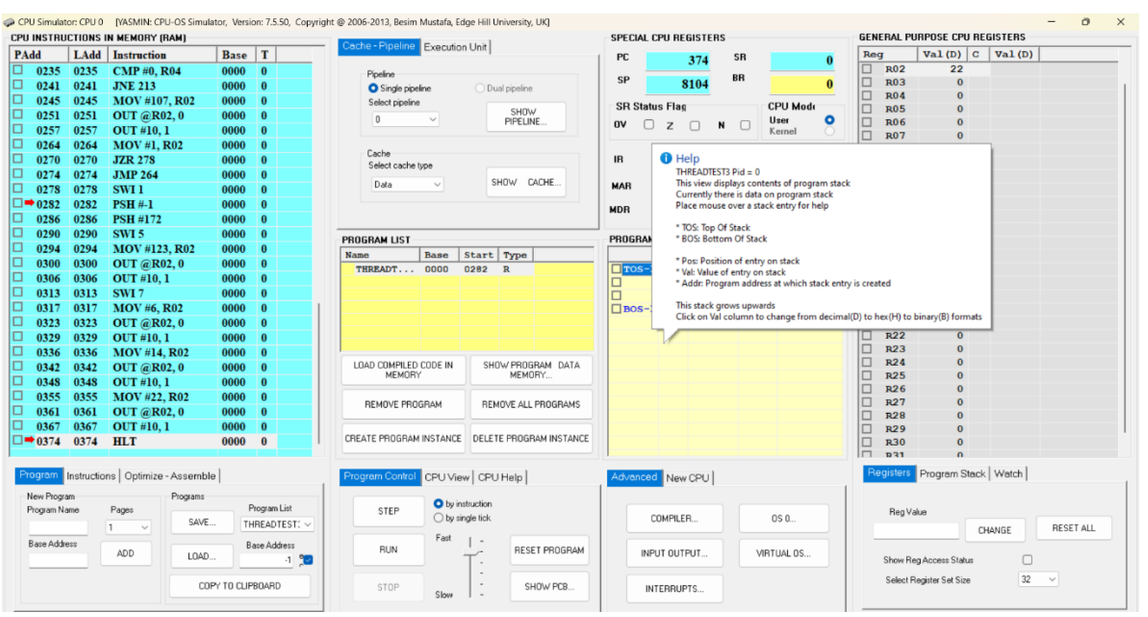
   Jawab : Thread-thread dalam suatu proses berbagi sumber daya dengan proses induknya karena mereka berada dalam ruang alamat yang sama. Ini berarti bahwa semua thread dalam satu proses memiliki akses ke memori yang dialokasikan untuk proses induk, termasuk variabel global, file yang dibuka, dan sumber daya lainnya seperti perangkat input/output. Berbagi sumber daya ini memungkinkan komunikasi yang lebih cepat antar-thread dibandingkan antar-proses karena tidak perlu menggunakan mekanisme komunikasi antar-proses yang berat. Namun, karena thread-thread berbagi memori, ada risiko terjadinya konflik atau race condition jika lebih dari satu thread mencoba mengakses atau memodifikasi data yang sama secara bersamaan tanpa pengaturan sinkronisasi yang tepat.

TUGAS

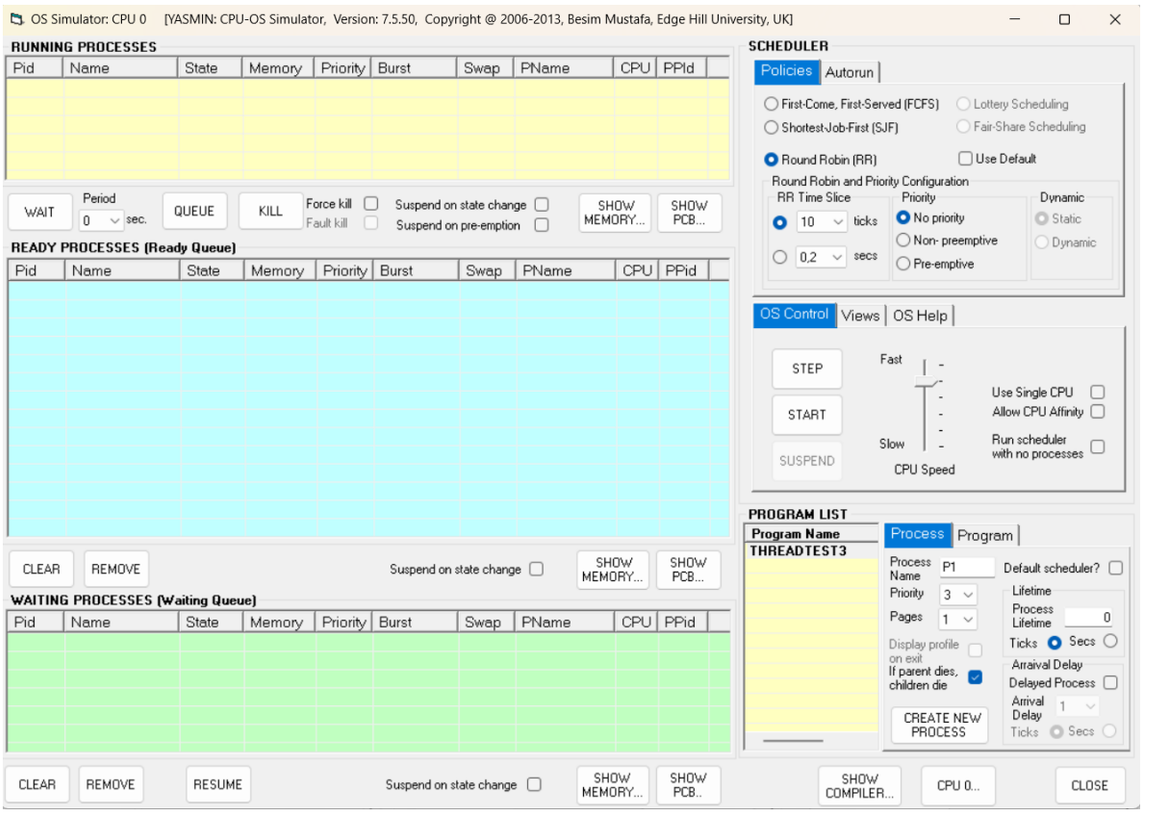
1. Buatlah program baru dengan memodifikasi source code threadtest2 (pada contoh program sebelumnya) dengan tahap tahap yang sesuai dan buatlah Kesimpulan.

    a. Ketikan source code berikut :

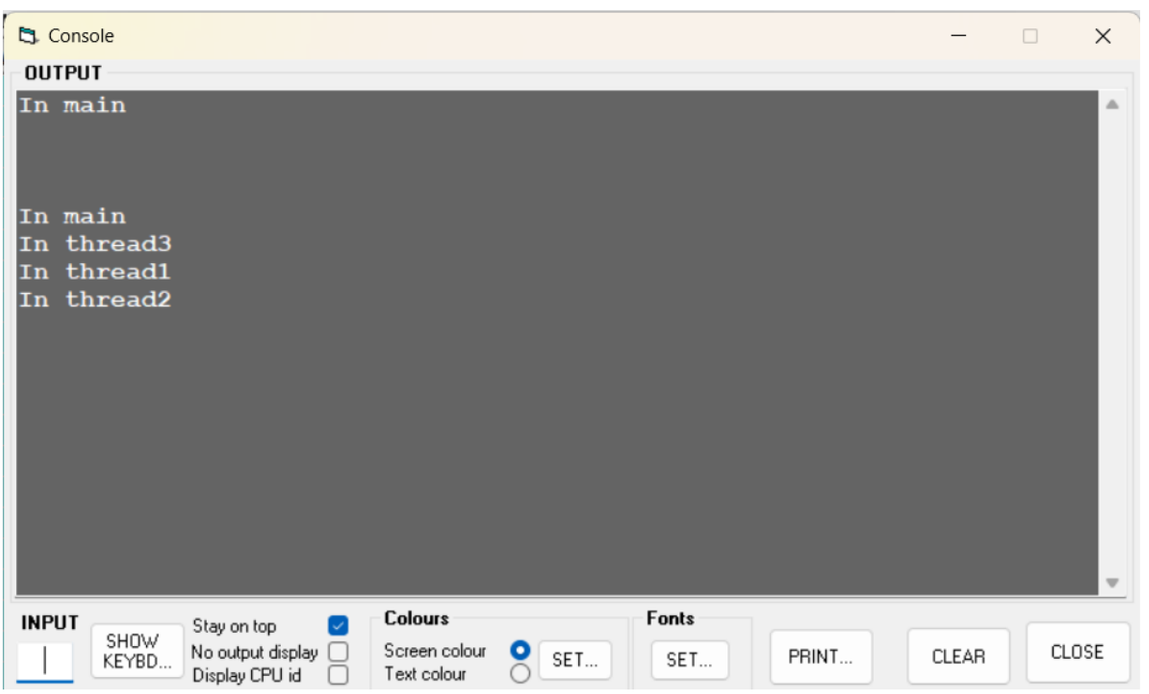
```
program ThreadTest3
  var s1 string (6)
  var s2 string (6)
  var s3 string (6)
  sub thread1 as thread
    s1 = "hello1"
    writeln("In thread1")
    while true
    wend
  end sub
  sub thread2 as thread
    s2 = "hello2"
    writeln("In thread2")
    while true
    wend
  end sub
  sub thread3 as thread
    call thread1
    call thread2
    s3 = "hello3"
    writeln("In thread3")
    while true
    wend
  end sub
  call thread3
  writeln("In main")
  wait
  writeln(s1)
  writeln(s2)
  writeln(s3)
end
```



b. Compile dengan menekan tombol COMPILE pada compiler, lalu run code berikut
c. Melalui CPU Simulator, tampilkan jendela konsol dengan klik tombol INPUT/OUTPUT... aktifkan STAY ON TOP.
d. Beralihlah ke jendela OS Simulator, buat satu proses dengan CREATE NEW PROCESS.
e. Pastikan jenis penjadwalan adalah ROUND ROBIN, 10 ticks dengan kecepatan simulasi maksimum.
f. Tekan tombol start.



g. Dan amatilah tampilan pada jendela console.



Di sini program saya di modifikasinya cuma menambahkan thread di mana terdapat tiga subrutin thread1, thread2, dan thread3. Setiap subrutin menulis pesan pada konsol untuk menandakan bahwa thread sedang berjalan, kemudian masuk ke dalam loop (while true wend). Ketika thread3 dipanggil oleh program utama, ia memulai thread1 dan thread2 sehingga ketiga

thread berjalan bersamaan. Program utama juga menuliskan "in main" setelah memanggil thread3, namun tidak bisa menampilkan nilai dari variabel s1, s2, dan s3 karena thread3 tidak pernah selesai akibat loop tak terbatas yang dimilikinya.

Kesimpulannya :

Program ini menunjukkan pentingnya pengaturan kontrol aliran dan kondisi penghentian dalam penggunaan thread. Karena setiap thread (thread1, thread2, dan thread3) masuk ke dalam loop tak terbatas, mereka tidak pernah selesai menjalankan tugasnya, yang menyebabkan program utama tidak bisa menampilkan nilai variabel s1, s2, dan s3 karena menunggu thread3 selesai. Hal ini mengakibatkan program terkunci dalam kondisi berjalan terus-menerus tanpa pernah mencapai akhir (deadlock). Dalam penerapan thread, sangat penting untuk memastikan bahwa setiap thread memiliki kondisi keluar yang jelas agar program dapat menyelesaikan seluruh instruksinya dan tidak berhenti di tengah jalan tanpa hasil.