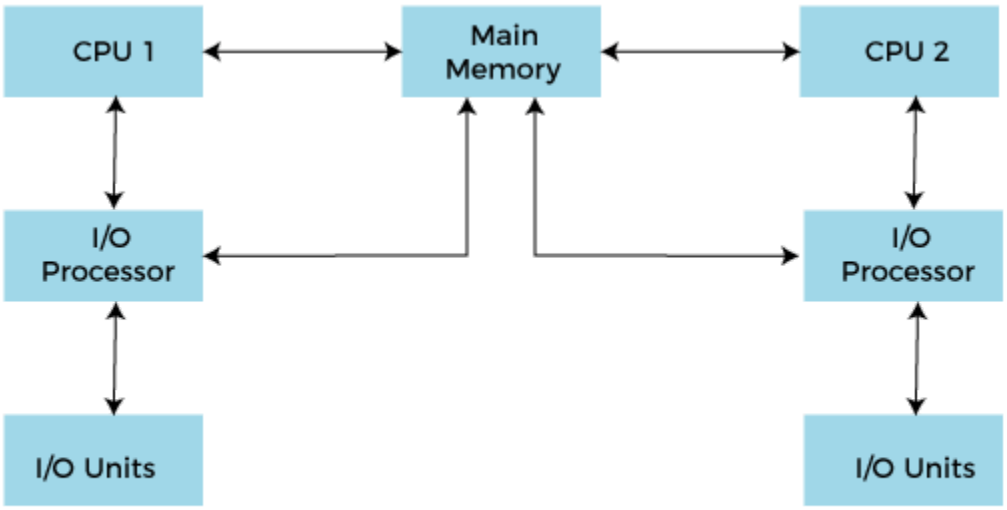
Nama :Iqbal Ramadhan Anniswa
NIM :21083010111
Matkul :Sistem Operasi A

MultiProcessing

Multiprocessing adalah penggunaan dua atau lebih unit pemrosesan pusat (CPU) dalam satu sistem komputer.Istilah ini juga mengacu pada kemampuan sistem untuk mendukung lebih dari satu prosesor atau kemampuan untuk mengalokasikan tugas di antara mereka. Ada banyak variasi pada tema dasar ini, dan definisi multiprosesor dapat bervariasi menurut konteksnya, sebagian besar sebagai fungsi dari bagaimana CPU didefinisikan (beberapa inti pada satu cetakan, beberapa cetakan dalam satu paket, beberapa paket dalam satu unit sistem, dll.).



Working of Multiprocessor System

Untuk menggunakan sistem operasi multiprosesor secara efektif, sistem komputer harus memiliki hal-hal berikut:

- A motherboard is capable of handling multiple processors in a multiprocessing operating system.
- Processors are also capable of being used in a multiprocessing system.

Keuntungan dari sistem operasi multiprocessing adalah:

- Peningkatan keandalan: Karena sistem multiprosesor, tugas pemrosesan dapat didistribusikan di antara beberapa prosesor. Ini meningkatkan keandalan

seolah-olah satu prosesor gagal; tugas dapat diberikan ke prosesor lain untuk diselesaikan.

- Meningkat di seluruh: Seiring bertambahnya beberapa prosesor, lebih banyak pekerjaan dapat dilakukan dalam waktu yang lebih sedikit
- Skala Ekonomi: Karena sistem multiprosesor berbagi periferal, perangkat penyimpanan sekunder, dan catu daya, mereka relatif lebih murah daripada sistem prosesor tunggal.

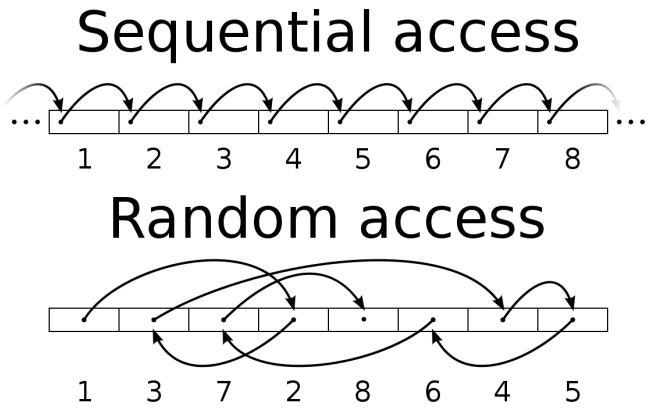
Kekurangan Sistem Operasi Multiprocessing:

- Sistem operasi multiprocessing lebih kompleks dan canggih karena menangani beberapa CPU secara bersamaan.

Di dalam Python tersebut ada package untuk bisa melakukan tersebut yaitu package multiprocessing.paket ini yang mendukung proses spawning menggunakan API yang mirip dengan modul threading. Paket multiprosesor menawarkan konkurensi lokal dan jarak jauh, yang secara efektif mengesampingkan Global Interpreter Lock dengan menggunakan sub proses alih-alih utas. Karena itu, modul multiprocessor memungkinkan pemrogram untuk sepenuhnya memanfaatkan banyak prosesor pada mesin tertentu. Ini berjalan di Unix dan Windows.

Disini saya memakai tiga time untuk melakukan multi processing

1. Sequential
   Sequential adalah dimana proses sebuah program dengan melakukan secara terurut



   Contoh pada pemprogram di python seperti di bawah ini

```
def odd_even(data):
    for i in range(1,data+1):
        if i%2 ==0:
            print(f'{i} Genap Get ID-{getpid()}')
        else:
            print(f'{i} Ganjil Get ID-{getpid()}')
```

✓ 0.3s

   Ini adalah fungsi yang kita jalankan

```
seq_start=time()
in_user=int(input('masukan data:'))
odd_even(in_user)
seq_akhir=time()
```

Ini adalah source code untuk sequential

```
seq_akhir-seq_start
✓ 0.8s
2.3874423503875732
```

ini adalah hasil process dalam Program

2. Process

Dalam multiprocessing, proses dimunculkan dengan membuat objek Process dan kemudian memanggil metode start()-nya. Proses mengikuti API dari threading.Thread. Contoh sepele dari program multi proses adalah

```
process_start=time()
memo=[]
for i in range(in_user):
    p=Process(target=odd_even,args=(i,))
    memo.append(p)
    p.start()
for sub in memo:
    p.join()

process_end=time()
process_end-process_start
✓ 0.2s
```

Source code untuk process

```
3 Ganjil Get ID-6976


4 Genap Get ID-69692 Genap Get ID-6986
2 Genap Get ID-6981
3 Ganjil Get ID-6964
5 Ganjil Get ID-6969
3 Ganjil Get ID-6986
4 Genap Get ID-6976
...

7 Ganjil Get ID-6981
8 Genap Get ID-69819 Ganjil Get ID-6986
```

```
0.2346353530883789
```

Ini waktu yang dibutuhkan

3. Pool Multi process
   Modul multiprocessing juga memperkenalkan API yang tidak memiliki analog dalam modul threading. Contoh utama dari hal ini adalah objek Pool yang menawarkan cara yang nyaman untuk memparalelkan eksekusi fungsi di beberapa nilai input, mendistribusikan data input ke seluruh proses (paralelisme data). Contoh berikut menunjukkan praktik umum dalam mendefinisikan fungsi tersebut dalam modul sehingga proses anak dapat berhasil mengimpor modul tersebut. Contoh dasar paralelisme data menggunakan Pool,

```
pool_awal=time()
pool=Pool()
pool.map(odd_even,range(1,in_user+1))
p.close()
pool_akhir=time()
pool_akhir-pool_awal
```

Source code untuk pool processing

```
6 Genap Get ID-71282 Genap Get ID-7130
3 Ganjil Get ID-7129
5 Ganjil Get ID-71273 Ganjil Get ID-71301 Ganjil Get ID-7128
...
8 Genap Get ID-7128
8 Genap Get ID-71309 Ganjil Get ID-712810 Genap Get ID-7127
```

Output pada pool multiprocessing

```
0.2842867374420166
```

Ini untuk waktu yang dibutuhkan

Kesimpulan
Bahwa waktu sequential dan process terakhir.sequential adalah membutuhkan waktu paling lama