**TRANSPOSE MATRIKS DENGAN CPU OS SIMULATOR**

Nama : Agista Rakhasanca

NIM : 4611419071

Prodi : Informatika

1. **Pendahuluan**

CPU OS Simulator merupakan sebuah tools untuk mensimulasikan proses-proses yang ada dalam CPU. CPU OS Simulator memungkinkan kita untuk mensimulasikan suatu instruksi pada CPU. Berkaitan dengan mensimulasikan instruksi CPU, CPU OS simulator juga mendukung bahasa assembly. Dengan fitur-fitur yang dimiliki CPU OS Simulator ini, memungkinkan kita untuk mensimulasikan program bahasa Assembly atau instruksi ke CPU. Dengan biegitu CPU OS simulator dapat membantu kita dalam mempelajari berbagai macam instruksi yang ada di CPU dengan UI yang mudah dipahami.

1. **Langkah Praktikum**

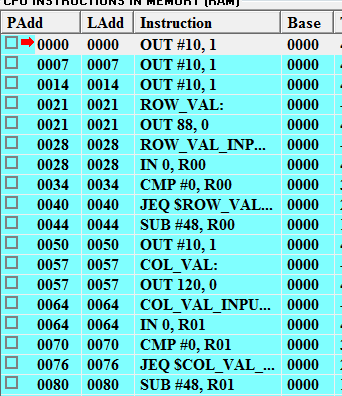
Pada Praktikum ini, saya akan membuat sebuah program transpose matrix menggunakan bahasa Assembly dalam CPU OS SImulator. Program ini nantinya akan memiliki fitur berupa user yang dapat melakukan input jumlah baris dan kolomnya sendiri, menginput matriks pada tiap baris dan kolom, menampilkan matriks yang diinput, kemudian menampilkan hasil transpose matriks.

1. **Persiapan**

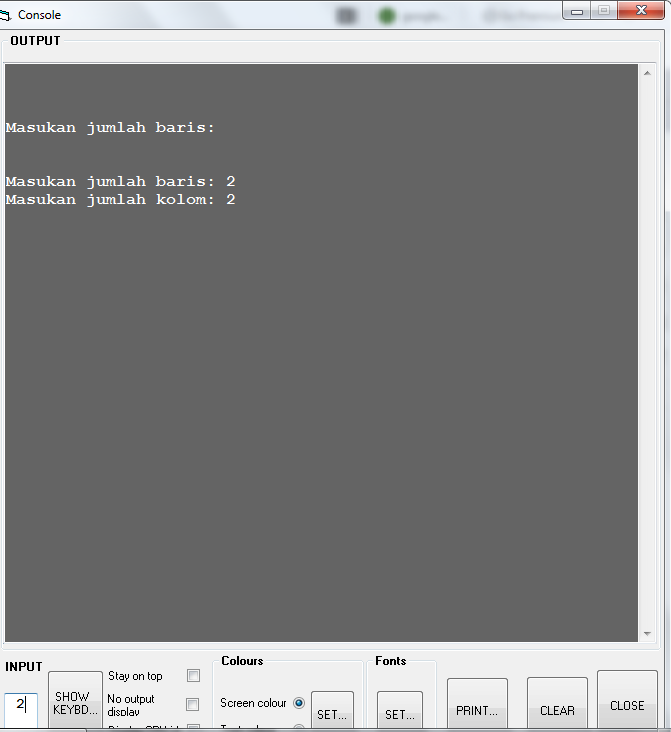
|  |  |  |  |
| --- | --- | --- | --- |
| No | Langkah | Penjelasan | Gambar |
| 1 | Membuat Program baru | Masukan nama program sesuai prefrensi masing-masing. Base Address yang saya gunakan adalah 0. | Persiapan1 |
| 2 | Membuka Input Output Console | Input Output ini nantinya akan digunakan untuk mengatasi input dari user dan akan menampilkan output ke user. Klik pada Input Output di menu advance, dan akan muncul Input Output Console. | Persiapan2Persiapan3 |
| 3 | Menambah Value ke Program Data Memory. | Nantinya program yang akan dibuat membutuhkan bantuan memory. Dimana memory nantinya akan berperan untuk menyimpan matriks tiap kolom dan baris, menyimpan string seperti “Masukan jumlah baris”.  Dengan cara klik “show Program Data” akan muncul menu untuk menambahkan value ke memory. | Persiapan4 |
| Kemudian lakukan penambahan data ke memory dengan memasukan value pada initialize data sesuai dengan tipe data yang diinginkan. Kemudian Klik update untuk menambah data. Setelah data ditambahkan, data tersbut dapat diakses dengan Logical address yang ada pada memory. | Persiapan 5 |
| Menambahkan data ke memory juga dapat dilakukan melalui script file SAS itu sendiri. Angka 7 diatas merepresentasikan banyaknya data yang ditampung. 2 angka sebelum data merepresentasikan Logical address dan Jumlah karakter. | Persiapan6 |

1. **Input Jumlah Baris dan Kolom**

Sekarang kita akan masuk kedalam langkah untuk membuat user bisa melakukan input utuk menentukan jumlah baris dan kolom yang akan digunakan. Program tersebut dibuat dengan insruksi dibawah ini:



Kemudian akan muncul output seperti gambar dibawah ini pada Input Output Console. Angka 2 untuk baris dan angka 2 untuk kolom adalah hasil input user.



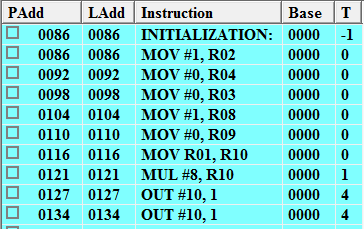
penjelasan tiap-tiap instruksi didalamnya:

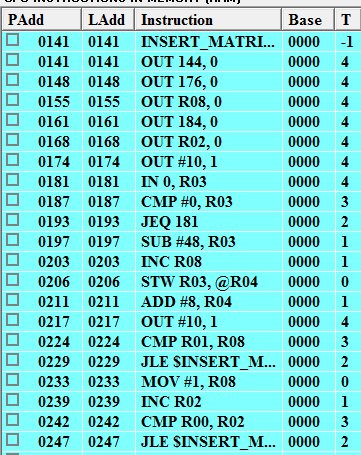
|  |  |
| --- | --- |
| OUT #10,1 | Digunakan untuk Membuat “Enter” Pada console, atau membuat loncat 1 baris pada console. |
| ROW\_VAL:  OUT 88,0 | ROW\_VAL merupakan label(bukan instruksi).  OUT 88,0 berarti melakukan output value dari memory dengan Logial address 88 dan page 0 ke IO Console. Perhatikan bahwa angka 88 tidak diikuti “#” yang berarti bahwa instruksi tersebut mengunakan mode Direct Memory. |
| ROW\_VAL\_INP | Label |
| IN 0, R00 | Memasukan value dari keyboard input kita ke register R00. Dalam hal ini keyboard value kita diarahkan pada value jumlah baris, sehingga R00 menyimpan value jumlah baris. |
| CMP #0,R00  JEQ $ROW\_VAL\_INP | Kedua instruksi ini bekerja secara sinergi. Dimana pertama kita membandingkan value input keboard dengan literal value 0. Jika value input keyboard dalam R00 adalah 0, yang berarti user belum melakukkan input, maka Fungsi JEQ akan dipanggil sehingga Instruksi akan lonca ke Label ROW\_VAL\_INP. Kedua instruksi ini dituukan agar program dapat menunggu user melakukan input. |
| SUB #48, R00 | Ini merupakan hal penting, karena setiap nilai input keyboard kita dimasukan ke register valuenya tidak sama dengan yang kita input. MIsal user melakukan input value 2, maka pada register value yang akan disimpan adalah 48 + value yang diinput, sehingga mengurangi R00 dengan 48 ditujukan untuk membuat register menyimpan value hasil input yang sebenarnya. |
| COL\_VAL | Pada label COL\_VAL hingga kebawah memiliki algoritma yang sama dengan ROW\_VAL diatas, hanya saja value yang diinput adalah baris ke R01. |

Berikut adalah

1. **Input Matriks Tiap Baris dan Kolom**

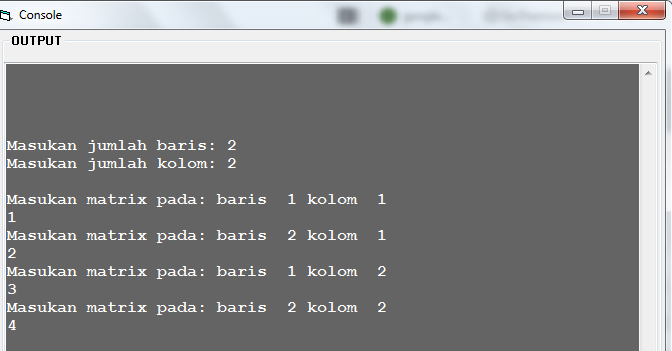
Langkah selanjutnya adalah membuat program agar user dapat melakukan input value pada matriks tiap baris dan kolom. Instruksinya sebagai berikut:





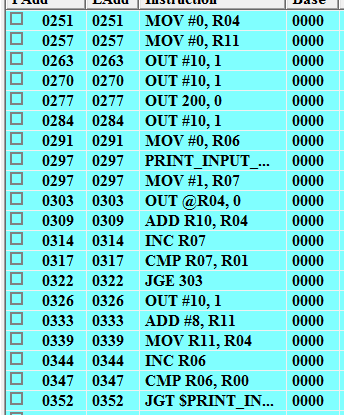
Algoritma Instruksi tersebut adalah sebagai beriktut:

1. Melakukan pendefinisian value pada bagian bawah label INITIALIZATION. Beberapa Register penting yang digunakan seperti R02 digunakan sebagai baris counter, R08 adalah kolom counter, dan R04 memiliki value yang merepresentasikan logical address memory,R08 digunakan untuk melomat ke value baris selanjutnya pada memory.
2. Memasukan value yang diinput dari keyboard ke register R03 yang berfungsi sebagai value hander.
3. Fungsi wait juga ditambahkan agar program akan menunggu user melakukan input. CMP #0, R03 dan JEQ merupakan bagian dari fungsi tersebut.
4. Melakukan increment pada R08 yang merupakan kolom counter dengan menggunakan instruksi INC. Increment dilakukan setiap user berhasil memasukan nilai ke R03.
5. Setelah melakukan INC pada R08, langkah selanjutnya adalah melakukan penyimpanan data matriks ke memory dengan instruksi SETW R03, @R04. Dimana insruksi tersebut akan menyimpan data ke logical address memory yang diakses secara indirect register mode pada R04.
6. Setelah data berhasil ditambah langkah selanjutnya adalah menambahkan 8 literal value ke R04. Alasannya adalah karena setiap bit word berukuran 8 bit maka dengan menambahkan value 8 pada R04 Logical address berpindah ke bitword selanjutnya.
7. Pengkodisian untuk melihat apakah jumlah kolom yang didefinisikan telah tercapai. Dilakukan dengan cara CMP R01,R08. JIka value R08 masih kurang dari sama dengan R01 maka instruksi akan looping kembali ke label INSERT\_MATRIX dengan Instruksi JLE.
8. Jika jumlah kolom terpenuhi maka langkah selanjutnya adalah melakukan INC pada baris counter yaitu R02 sehingga Dilakukan kembali pengkondisian apakah value R02 sudah mememnuhi R00, jika belum terpenuhi maka looping akan terjadi kembali.
9. Sehingga inputnya sebagai berikut.



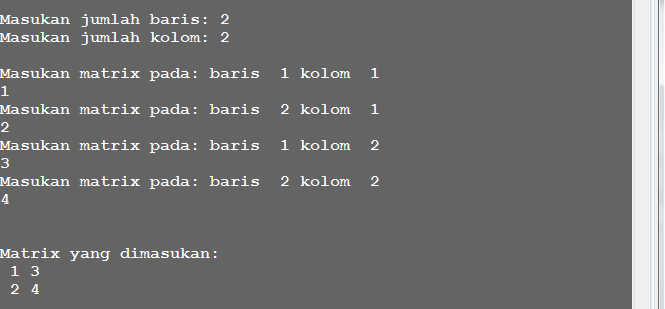
1. **Menampilkan Matriks yang Diinput**

Langkah selanjutnya adalah menampilkan matriks yang diinput oleh user dengan IO console.



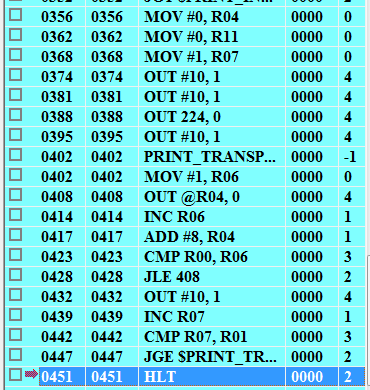
Algoritma Insruksi diatas adalah sebagai berikut:

1. Melakukan inisialisasi value ke register. Contohnya seperti R06 yang merupakan baris counter, R07 merupakan kolom counter. R11 merupakan Handler dari value memory R04, Kemudian pendefinisian ulang value R04 agar menjadi
2. Langkah untuk melakukan output dari matriks yang telah dibuat adalah dengan menggunakan instruksi OUT @04,0. Instruksi tersebut digunakan untuk melakukan print ke IO console. Value diambli dari memory address yang value nya disimpn dalam R04.
3. Lagnkah selanjutnya adalah dengan menambahkan value R04 dengan R10, value R10 adalah jumlah baris \* #8, dimana 8 merupakan jumlah bitword tiap memory address. Fungsi dari R10 adalah untuk melompat ke kolom selanjutnya, karena jika hanya melakukan penambahan value #8 ke R04 hanya akan melompat ke baris selanjutnya bukan ke kolom selanjutnya.
4. Melakukan INC(increment) pada R07, karena R07 adalah kolom counter.
5. Jika value R07 <= R01 maka program akan looping ke address 303, dimana instruksi menampilkan matriks dilakukan.
6. JIka value R07 > R01 maka program akan lanjut ke instruksi selanjutnya.
7. Jika program lanjut ke instruksi selanjutnya maka itu artinya baris pertama berhasil ditulis sehingga sekarang kita menuju baris ke 2.
8. Untuk menuju baris ke 2 lakukan Increment(INC) pada R06 yang merupakan baris counter.
9. Setelah menuju ke baris 2, kita lakukan add #8 ke R04 agar melompat ke baris selanjutnya pada value yang kita store pada memory.
10. Jika R00 > R06 maka Instruksi akan kembali ke atas dimana langka ke 2 dilakukan.
11. JIika R06>R00 maka instruksi akan selsesai dan menuju ke instruksi selanjutnya dan matriks berhasil ditampilkan.
12. Berikut adalah output dari Instriksi untuk menampilkan matriks.



1. **Menampilkan Hasil Transpose Matriks**

Lagnkah terakhir adalah untuk menampilkan Transpose Matriks.

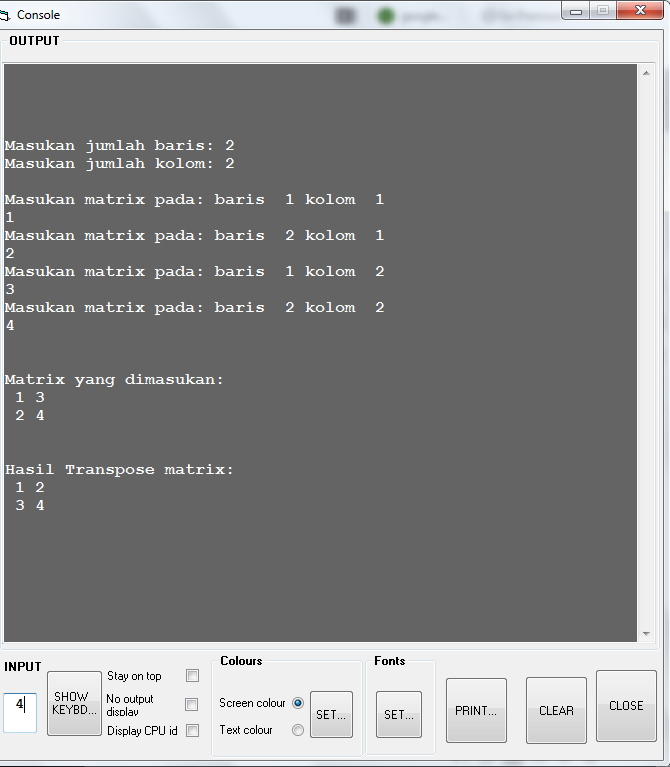


Sebenarnya pada dasarnya Algoritma untuk menampilkan matriks dan menampilkan matriks transpose bisa dibilang hampir sama. Perbedaanya, adalah pada bagaimana memory dibaca. Pada Instruksi untuk menampilkan matriks, kita malkukan penambahan value pada R04(register yang menampung logial address memory), dengan R10(baris\*8) yang berarti kita membaca value kolom selanjutnya bukan beris selanjutnya. Untuk membaca baris selanjutnya kita hanya berlu melakukan ADD #8,R04.

Nah, Dari uraian diatas maka dapat diperoleh algoritma untuk menampilkan transpose matriks.

1. Menggunakan algoritma yang sama untuk melakukan print ke IO console pada instruksi untuk menampilkan matriks. Hal ini berarti program akan ditulis dari kolom terlebih dahulu.
2. Membaca memory dengan melompat ke bitword selanjutnya dengan ADD #8,R04. Hal ini mengakibatkan program akan ditulis dengan penulisn kolom, namun value yang dimasukkan adalah value baris, sehingga transpose matriks terbentuk.

Berikut adalah output:



1. **Penutup**

Sekian dari saya tentang bagaimana membuat program Transpose matriks pada CPU OS SImulator.

1. **Lampiran**

Video Youtube : <https://youtu.be/mRSZZps-Y40>

Github : <https://github.com/agista69/transposeMatrixAssembly>