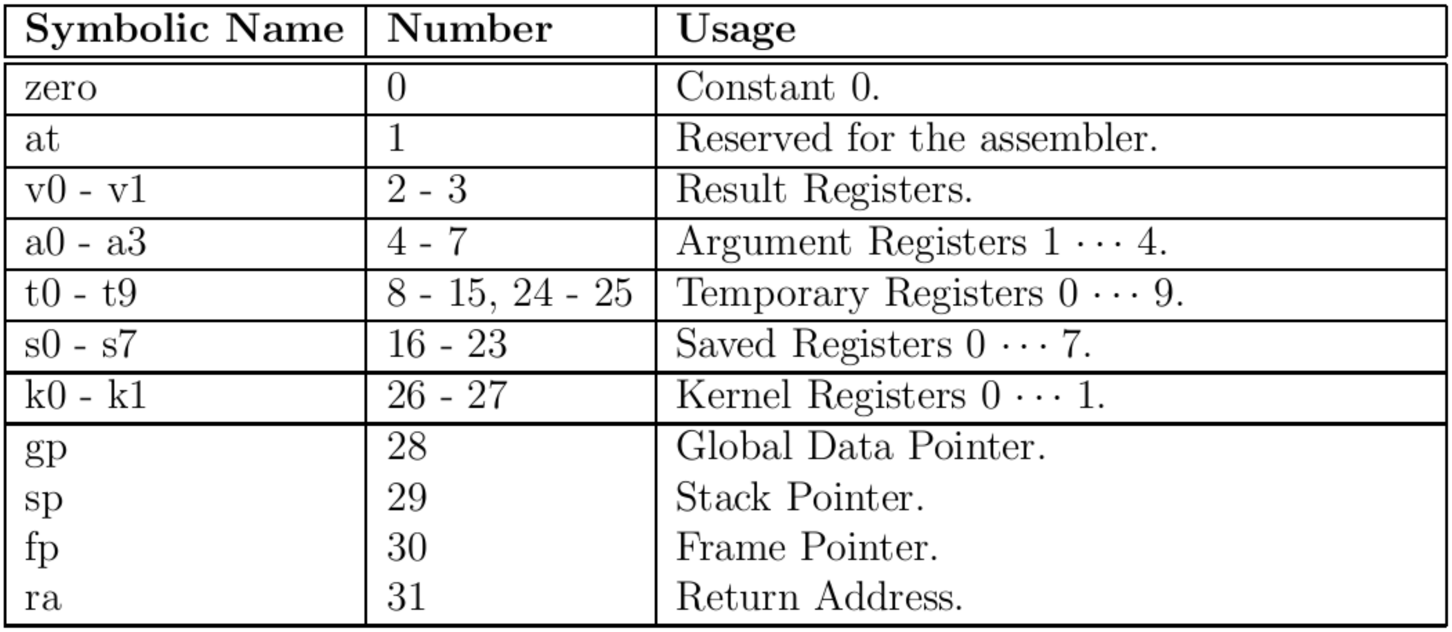
কোড লিখন

কোড করার আগে কিছু বিষয় জেনে নেওয়া যাক।

রেজিস্টারে নিয়ে কাজ করাঃ

সাধারণত আমরা যখন কোড লিখি তখন ভেরিয়েবল ডিক্লেয়ার করি। বিভিন্ন কাজের জন্য ভিন্ন ভিন্ন ভেরিয়েবল রয়েছে। কিন্তু SPIM এ কোড লিখার ক্ষেত্রে ভেরিয়েবল ডিক্লেয়ার করা যায় না। MIPS নামক ইনস্ট্রাকশন সেট (কিছু নির্দেশ) থেকে ইনস্ট্রাকশন নিয়ে কাজ করা হয়। সবচেয়ে গুরুত্বপূর্ণ হল, এখানে প্রসেসর এর নিজস্ব মেমরি যা রেজিস্টার নামে পরিচিতে, তা নিয়ে কাজ করা হয়। অর্থাৎ যখন কোন ডাটা নিয়ে আমরা কাজ করি, তখন তা রেজিস্টারে নিয়ে কাজ করা হয়। MIPS ইনস্ট্রাকশন সেট নিয়ে কাজ করার ক্ষেত্রে আমরা ৩২ টি রেজিস্টার নিয়ে কাজ করতে পারি। নিচে এই ৩২ টি রেজিস্টারের সংক্ষিপ্ত পরিচিতি দেয়া হলঃ



Symbolic Name	Number	Usage
zero	0	Constant 0.
at	1	Reserved for the assembler.
v0 - v1	2 - 3	Result Registers.
a0 - a3	4 - 7	Argument Registers 1 · · · 4.
t0 - t9	8 - 15, 24 - 25	Temporary Registers 0 · · · 9.
s0 - s7	16 - 23	Saved Registers 0 · · · 7.
k0 - k1	26 - 27	Kernel Registers 0 · · · 1.
gp	28	Global Data Pointer.
sp	29	Stack Pointer.
fp	30	Frame Pointer.
ra	31	Return Address.

কিন্তু সাধারণত তিন ধরনের রেজিস্টার দিয়েই আমরা কাজ করে থাকি। এগুলো হলঃ

- রেজাল্ট রেজিস্টার (v0, v1)
- আর্গুমেন্ট রেজিস্টার (a0 , a1, a2, a3)
- টেম্পোরারি রেজিস্টার (t0, t1, t2, t3, t4, t5, t6, t7, t8, t9)

ভিন্ন ভিন্ন সেগমেন্টে কাজ করাঃ

পুরো কোডিং সাধারণত দুটি সেগমেন্টে সম্পূর্ণ করা হয়।

- টেক্সট সেগমেন্টে (মূল কোডিং এই অংশে থাকে)
- ডাটা সেগমেন্টে (বিভিন্ন লেবেল ও তার ভ্যালু এই অংশে থাকে)

এক্সটেনশন ঠিক রাখাঃ

প্রতিটি কোড এর এক্সটেনশন হবে “.asm” . “.asm” হল “Assembly” এর সংক্ষিপ্ত রূপ। ফাইলের নামকরণের সাধারণ স্ট্রাকচার হলঃ

NameofFile.extension

NameofFile.asm

অর্থাৎ, কোন ফাইলের নাম যদি হয় “HelloWorld”, তাহলে এক্সটেনশন সহ এর নাম হবে:

HelloWorld.asm

এবার একটা ছোট্ট প্রোগ্রাম দেখা যাক, যার মূল কাজ হল Console -এ

Hello Bangladesh - প্রিন্ট করে দেখাবে।

কিভাবে ফাইল সেভ করবঃ

এর জন্যে প্রথম Notepad চালু করি। এবার তাতে নিম্নোক্ত কোড লিখি এবং সেভ করি।

সেভ করার জন্য প্রথমে, File > Save -এ ক্লিক করি।

এরপর, একটি ডায়ালগ বক্স ওপেন হবে। সেখানে, File name এর জায়গায়,

*.txt -কে ডিলিট করে, HelloWorld.asm লিখি এবং Save করি।

HelloWorld.asm - কোডঃ

```
#HelloWorld.asm
.text

main:
la  $a0, hello_msg
li  $v0, 4
syscall
li  $v0, 10
syscall

   ##Data for the program

   .data
   hello_msg:    .asciiz "Hello Bangladesh"

   ##end HelloWorld.asm
```

এবার পুরো কোডটাকে লাইন বাই লাইন দেখা যাক

```
1. #HelloWorld.asm
```

এই লাইনে আসলে কমেন্ট লিখা হয়েছে। যে কোন কমেন্ট লিখতে চাইলে, তা “#” দিয়ে শুরু হয়। এবং কম্পাইলার এই কমেন্টকে এক্সিকিউট করে না, অর্থাৎ সে কমেন্টকে ইগনোর করে। ইগনোর করার মানে হল, সে একে নিয়ে কোন কাজ করবে না।

আরেকটু সহজ ভাবে ভাবি, অনেক সময় তো এরকম হয়েই থাকে, কেউ হয়তো তো আপনাকে কিছু বলছে, কিন্তু আপনি হয়তো তার কথা পাত্তাই দিচ্ছেন না, হয় না?? মাঝে মাঝেই হয়। সবার কথা তো আর গুরুত্ব দিয়ে শোনার কিছু নেই ;-P

অনেক ক্ষেত্রেই আমরা এক কান দিয়ে কথা ঢুকাই আর আরেক কান দিয়ে তা বের করে দেই। কমেন্টিং এর ক্ষেত্রেও তাই হয়, কম্পাইলার একে পাত্তাই দেয় না :-D

এখানে আমি কোডিং এর শুরুতেই আমার কোডটির নাম লিখেছি। কমেন্ট এর স্ট্রাকচার অর্থাৎ লেখার নিয়ম হলঃ

```
# comment you want to write
```

প্রথমে “#” লেখতে হয়, এরপর কমেন্ট হিসেবে যা লিখতে চাই তা লিখতে হবে।

কেউ যদি কমেন্ট লিখতে চায়, “written by me” তাহলে তা হবে:

```
# written by me
```

```
2. .text
```

শুরুতেই বলা হয়েছে যে, পুরো কোডিং সাধারণত দুটো সেগমেন্টে করা হয়। “.text” -দিয়ে বোঝানো হচ্ছে যে, এখান থেকে আমার কোড শুরু হচ্ছে।

```
3.
```

এই লাইনে কিছু লেখা হয় নি। প্রোগ্রামার ইচ্ছে করলেই, যে কোন লাইন খালি রাখতে পারে। এতে কোন সমস্যা হয় না। বরং কোড ঠিকমত পড়তে সুবিধা হয়। যদি কোন লাইন খালি রাখা না হয় আর কোডও অনেক বড় হয়ে যায়, তখন প্রতিটি লাইন আলাদা করতে অসুবিধা হয়। সাধারণত পড়ার সুবিধার জন্যই লাইন খালি রাখা হয়।

```
4. main:
```

প্রতিটি প্রোগ্রাম একটি নির্দিষ্ট জায়গা থেকে শুরু হয়। আর MIPS - এর ক্ষেত্রে তা শুরু হয় “main : ” -অংশ থেকে। সুতরাং এখান থেকেই আমার মূল প্রোগ্রাম চালু হল। এটা হল একটি প্রোগ্রামের জন্য স্টার্টিং পয়েন্ট। “main : ” অংশটুকু ছাড়া কোন প্রোগ্রাম চলতে পারে না। সুতরাং “main : ” অবশ্যই কোডে থাকতে হবে।

```
5. la   $a0, hello_msg
```

$a0 রেজিস্টারে hello_msg লেবেলের Address লোড করতে বলা হচ্ছে। যদি কোডের ১৪ নাম্বার লাইন দেখি, তাহলে দেখতে পাব, hello_msg: লেবেলের বিপরীতে “Hello Bangladesh” লেখা রয়েছে।

```
14.    hello_msg:    .asciiz "Hello Bangladesh"
```

la - এর পূর্ণ রূপ হলঃ Load Address. এর কাজ হল, $a0 রেজিস্টারে কোন লেবেলের address রেখে দেয়া। যেন পরে যদি ঐ লেবেল দরকার হয়, ঐ লেবেলে যে ভ্যালু রাখা আছে বা যা লিখা আছে, তা নিয়ে আসতে পারে। এর স্ট্রাকচার হলঃ

la destination, label_address

la $a0, hello_msg

destination - বলতে বোঝায়, যেখানে/যে রেজিস্টারে রাখা হবে তার নাম।

অর্থাৎ, $a0 - রেজিস্টারে hello_msg -লেবেলের আড্রেস লোড করা হয়েছে।

```
6. li   $v0, 4
```

li - এর পূর্ণরূপ হল Load Constant. অর্থাৎ -এর সাহায্যে রেজিস্টারে কোন একটি ভ্যালু রাখা হয় / লোড করা হয়। এখানে $v0 রেজিস্টারে একটি ভ্যালু 4 লোড করতে বলা হচ্ছে। এর স্ট্রাকচার হলঃ

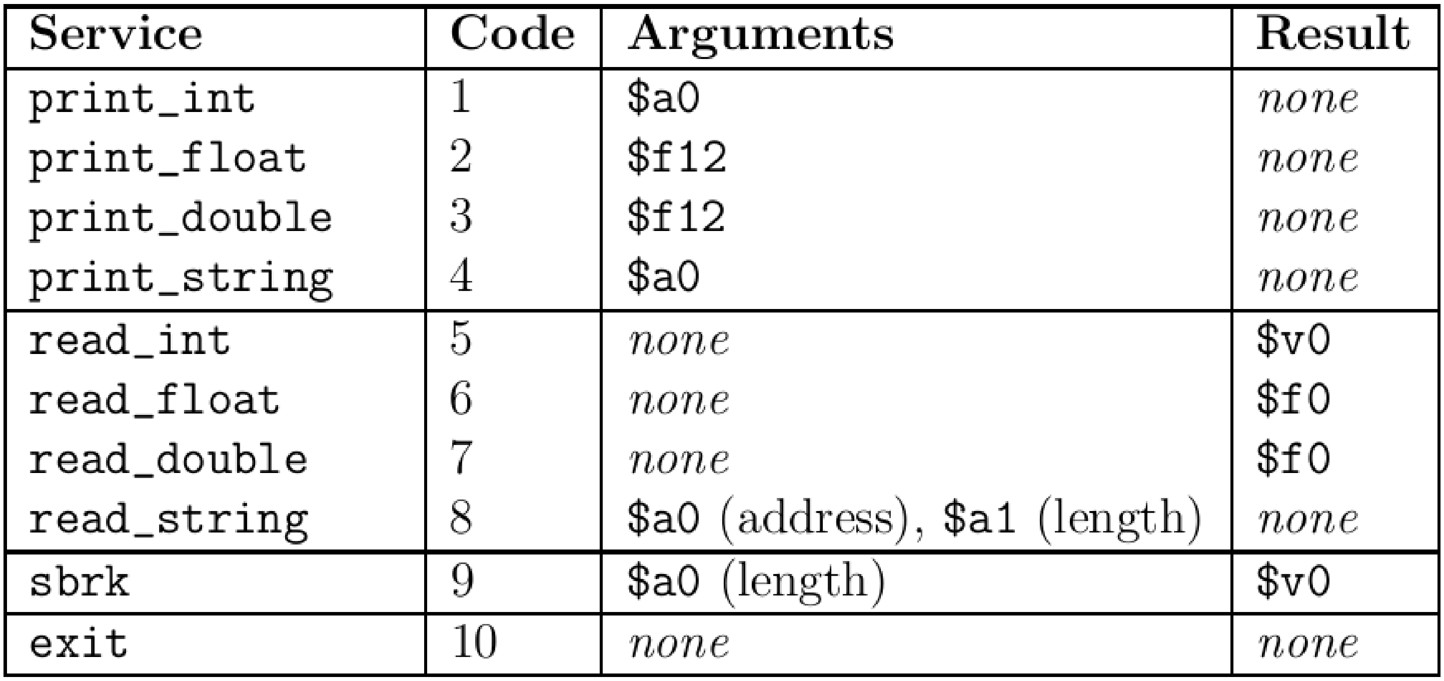
li destination, value

li $v0, 4

এখানে $v0 রেজিস্টারে 4 লোড করা হয়েছে।

7. syscall

syscall -এর পূর্ণরূপ হল : system Call. অর্থাৎ কোন একটি কাজ করে দেয়ার জন্য সিস্টেম কে কল করা হচ্ছে। সহজ কথায় বলতে গেলে, এটি কিছু কিছু কাজ করার যেমনঃ Console -এ কোন কিছু প্রিন্ট করা অথবা ইউজারের কাছ থেকে কোন ভ্যালু নেয়া ইত্যাদি কাজের সমন্বয় করে থাকে। syscall কি কি কাজ করতে পার, নিচে তার একটি ছক দেখান হলঃ



Service	**Code**	**Arguments**	**Result**
print_int	1	$a0	*none*
print_float	2	$f12	*none*
print_double	3	$f12	*none*
print_string	4	$a0	*none*
read_int	5	*none*	$v0
read_float	6	*none*	$f0
read_double	7	*none*	$f0
read_string	8	$a0 (address), $a1 (length)	*none*
sbrk	9	$a0 (length)	$v0
exit	10	*none*	*none*

এখানে দেখা যাচ্ছে, syscall এর ১০ টি কোড রয়েছে। প্রতিটি কোডের ভিন্ন ভিন্ন কাজ রয়েছে। যেমন:

কোড ১ : কোন ইন্টিজার ভ্যালু প্রিন্ট করা। (-১,-২০০, ০, ১, ৪, ১৯৩৪ এগুলো হল ইন্টিজার সংখ্যা যা পূর্ণ সংখ্যা নামে বাংলায় পরিচিত)

কোড ৪: স্ট্রিং প্রিন্ট করে।

কোড ৫: ইউজারের কাছে থেকে ইন্টিজার ভ্যালু নেয়।

কোড ১০: প্রোগ্রাম টার্মিনেশনের কাজে ব্যবহৃত হয়।

একটি বিষয় মনে রাখা জরুরী, syscall যখনি করা হোক না কেন, syscall করার সাথে সাথে , syscall রেজিস্টার $v0 -তে যেয়ে দেখে তার ভ্যালু কত। $v0 -এর ভ্যালু যত হবে, syscall তার নিজস্ব কোড অনুযায়ী সেই কাজ করবে। এজন্যেই,

লাইন #7 - এ যখন syscall করা হল, syscall - $a0 -তে থাকা hello_msg

```
5. la  $a0, hello_msg
6. li  $v0, 4
7. syscall
```

লেবেলের Address এ যাবে এবং তাতে থাকা ভ্যালু “Hello Bangladesh” -প্রিন্ট করবে console - এ। কারণ syscall এর জন্য কোড 4: এর মানে হল, স্ট্রিং প্রিন্ট করতে বলা হচ্ছে।

এই কারণেই, প্রথমে $v0 -তে একটি ভ্যালু 4 -সেট করা হয়েছে এবং তারপর syscall - কে ডাকা হয়েছে। ফলে syscall প্রথমে নির্দেশ পায় স্ট্রিং প্রিন্ট করার, এবং পরে syscall চলে যায় $a0 রেজিস্টারে দেখতে সেখানে কি আছে। $a0 রেজিস্টারে hello_msg লেবেল খুঁজে পায়। এখন সে hello_msg লেবেলটি খুঁজতে

শুরু করে এবং কোডের ১৪ নং লাইনে তা খুঁজে পায়।

```
14.    hello_msg:    .asciiz "Hello Bangladesh"
```

তারপর লাইন #১৪ -এ `"Hello Bangladesh"` স্ট্রিং দেখে এবং তা console -এ প্রিন্ট করে।

কোন কারণে যদি, syscall -এ এমন কোন ভ্যালু সেট করা হয় যা 1-10 এর মাঝে নেই, তাহলে syscall কোন কাজ করতে পারবে না, এবং সিমুলেটর প্রোগ্রাম সঠিকভাবে আউটপুট প্রদর্শন করতে পারবে না।

```
8. li   $v0, 10

9. syscall
```

লাইন ৮: $v0 রেজিস্টারে ভ্যালু 10 লোড করা হয়েছে।

লাইন ৯: syscall -করা হয়েছে। যেহেতু 10-এর মানে প্রোগ্রাম টার্মিনেশন, তাই স্বাভাবিকভাবে প্রোগ্রাম syscall প্রোগ্রাম টার্মিনেট করবে, অর্থাৎ প্রোগ্রাম থেকে বের হয়ে যাবে।

এখন প্রশ্ন হতেই পারে, যদি লাইন # ৯ তেই, প্রোগ্রামের টার্মিনেশন হয়, তাহলে পরের লাইন গুলোর কি হবে ?

আচ্ছা, আপনার একটু ভাবতে থাকুন, এই সময়ে আমি পরের লাইন গুলোতে কি ঘটনা ঘটছে তা একটু বোঝানোর চেষ্টা করি।

```
11.    ##Data for the program
```

আরেকটি কমেন্ট। এই লাইন থেকে আমার প্রোগ্রামের ডাটা সেগমেন্ট এর কথা বলা রয়েছে। যেহেতু এটি একটি কমেন্ট, তাই কম্পাইলার একে ইগনোর করবে।

```
13.     .data
14.     hello_msg:    .asciiz "Hello Bangladesh"
```

লাইন ১৩: .data - এই অংশ থেকে প্রোগ্রামের ডাটা সেগমেন্ট অংশে কি করা হবে তা বলা হয়েছে। যেহেতু .text সেগমেন্ট থেকে .data সেগমেন্ট আলাদ, তাই আগে .text সেগমেন্টের সব কথা বলা হয়েছে, তারপর .data -সেগমেন্টে কাজ শুরু হল।

লাইন ১৪:

```
14.     hello_msg:    .asciiz "Hello Bangladesh"
```

hello_msg: -- এই অংশে hello_msg নাম লেবেলটি লেখা হয়েছে।

.asciiz -- hello_msg এ থাকা স্ট্রিং কি ধরনের তা বলা হয়েছে। । .asciiz - দ্বারা বোঝানো হচ্ছে যে, স্ট্রিংটি ASCII ক্যারেক্টার দিয়ে তৈরী হয়েছে, এবং এর শেষে একটি Null Terminator রয়েছে, যা নির্দেশ করে যে, এখানে অর্থাৎ Null Terminator - এ এসে, স্ট্রিংটি শেষ হয়েছে। সহজ কথায়, যখন SPIM - এই অর্থাৎ Null Terminator এ আসে তখন সে বুঝতে পারে, স্ট্রিংটি এখানেই শেষ হয়ে গেছে, এরপর আর কিছু নেই।

“Hello Bangladesh” - এটি হল সেই স্ট্রিংটি যা hello_msg লেবেলে সংরক্ষিত রয়েছে।

```
##end HelloWorld.asm
```

এটি আরেকটি কমেন্ট। এখানে বলা হচ্ছে, এই লাইনে এসেই আমাদের কোডিং শেষ হয়েছে।

কমেন্ট করতেই হবে এমন কোন ধরা বাঁধা নিয়ম নেই। তবে করলেই পরবর্তিতে কোডের কোথায় কি আছে/ কেন কি লিখা হয়েছে তা বুঝতে সমস্যা হয়। বিশেষত যখন অনেক বড় কোন কোড লিখা সে ক্ষেত্রে কমেন্টিং খুব উপকারী।