

Thực hành Kiến trúc máy tính

Giảng viên: Nguyễn Thị Thanh Nga
Khoa Kỹ thuật máy tính
Trường CNTT&TT

Mục tiêu

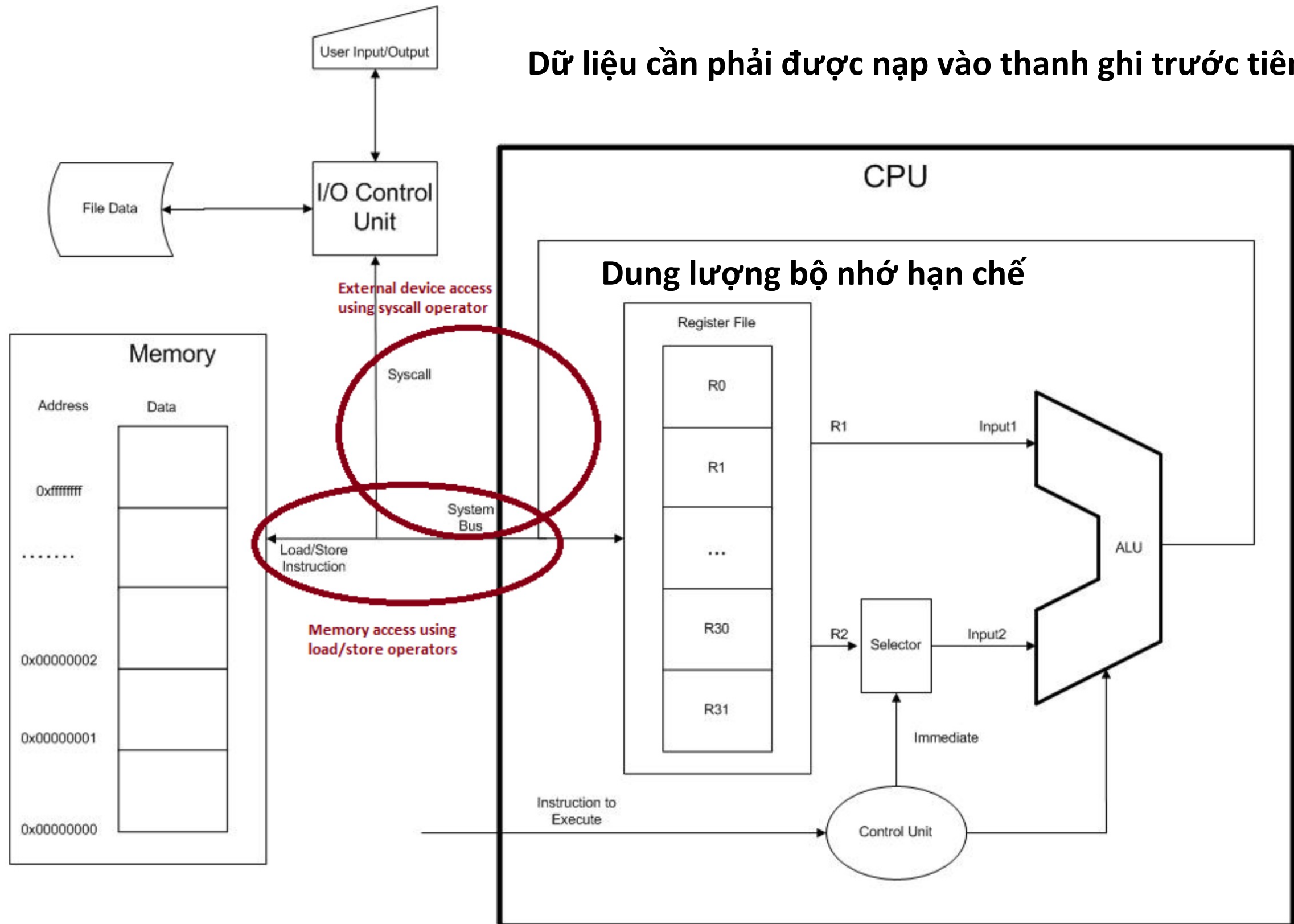
- Nguyên lý cơ bản về tập lệnh của bộ xử lý MIPS.
- Sử dụng được các lệnh hợp ngữ cơ bản và sử dụng công cụ gỡ rối để kiểm nghiệm lại các kiến thức về tập lệnh và hợp ngữ.
- Thành thạo với các chỉ thị biên dịch (Directives) để công cụ MARS có thể dịch hợp ngữ thành mã máy một cách chính xác.

Nội dung

- Bộ nhớ và các loại thanh ghi
- Cách chú thích một chương trình
- Một số chỉ thị hợp ngữ **.text**, **.data**, **.ascii**, **.space** và **.word**
- Nhãn
- Một số toán tử như **li**, **la**, **lw** và **move**
- Một số dịch vụ hệ thống để giao tiếp với giao diện người dùng: dịch vụ 1, 4, 5 và 8.
- Phân biệt giữa tham chiếu và tham trị.

Kiến trúc MIPS CPU

Dữ liệu cần phải được nạp vào thanh ghi trước tiên



Thanh ghi

\$zero (\$0):

- Dành cho mục đích sử dụng đặc biệt
- Luôn chứa giá trị 0
- Có thể được đọc nhưng không thể ghi.

Mnemonic	Number	Mnemonic	Number	Mnemonic	Number
\$zero	\$0		\$t3	\$11		\$s6	\$22
\$at	\$1		\$t4	\$12		\$s7	\$23
\$v0	\$2		\$t5	\$13		\$t8	\$24
\$v1	\$3		\$t6	\$14		\$t9	\$25
\$a0	\$4		\$t7	\$15		\$k0	\$26
\$a1	\$5		\$s0	\$16		\$k1	\$27
\$a2	\$6		\$s1	\$17		\$gp	\$28
\$a3	\$7		\$s2	\$18		\$sp	\$29
\$t0	\$8		\$s3	\$19		\$fp	\$30
\$t1	\$9		\$s4	\$20		\$ra	\$31
\$t2	\$10		\$s5	\$21			

Thanh ghi

\$at (\$1):

- Dành cho bộ hợp ngữ.
- Không dành cho người lập trình sử dụng.

Mnemonic	Number	Mnemonic	Number	Mnemonic	Number
\$zero	\$0		\$t3	\$11		\$s6	\$22
\$at	\$1		\$t4	\$12		\$s7	\$23
\$v0	\$2		\$t5	\$13		\$t8	\$24
\$v1	\$3		\$t6	\$14		\$t9	\$25
\$a0	\$4		\$t7	\$15		\$k0	\$26
\$a1	\$5		\$s0	\$16		\$k1	\$27
\$a2	\$6		\$s1	\$17		\$gp	\$28
\$a3	\$7		\$s2	\$18		\$sp	\$29
\$t0	\$8		\$s3	\$19		\$fp	\$30
\$t1	\$9		\$s4	\$20		\$ra	\$31
\$t2	\$10		\$s5	\$21			

Thanh ghi

\$v0-\$v1 (\$2-\$3)

- Lưu giá trị trả về của các chương trình con.
- \$v0 cũng được sử dụng để gọi syscall.

Mnemonic	Number	Mnemonic	Number	Mnemonic	Number
\$zero	\$0		\$t3	\$11		\$s6	\$22
\$at	\$1		\$t4	\$12		\$s7	\$23
\$v0	\$2		\$t5	\$13		\$t8	\$24
\$v1	\$3		\$t6	\$14		\$t9	\$25
\$a0	\$4		\$t7	\$15		\$k0	\$26
\$a1	\$5		\$s0	\$16		\$k1	\$27
\$a2	\$6		\$s1	\$17		\$gp	\$28
\$a3	\$7		\$s2	\$18		\$sp	\$29
\$t0	\$8		\$s3	\$19		\$fp	\$30
\$t1	\$9		\$s4	\$20		\$ra	\$31
\$t2	\$10		\$s5	\$21			

Thanh ghi

\$a0-\$a3 (\$4-\$7)

- Được sử dụng để truyền đối số (hoặc tham số) vào các chương trình con.

Mnemonic	Number	Mnemonic	Number	Mnemonic	Number
\$zero	\$0		\$t3	\$11		\$s6	\$22
\$at	\$1		\$t4	\$12		\$s7	\$23
\$v0	\$2		\$t5	\$13		\$t8	\$24
\$v1	\$3		\$t6	\$14		\$t9	\$25
\$a0	\$4		\$t7	\$15		\$k0	\$26
\$a1	\$5		\$s0	\$16		\$k1	\$27
\$a2	\$6		\$s1	\$17		\$gp	\$28
\$a3	\$7		\$s2	\$18		\$sp	\$29
\$t0	\$8		\$s3	\$19		\$fp	\$30
\$t1	\$9		\$s4	\$20		\$ra	\$31
\$t2	\$10		\$s5	\$21			

Thanh ghi

\$t0-\$t9 (\$8-\$15, \$24-\$25)

- Được sử dụng để lưu các giá trị tạm thời.
- Giá trị của các biến tạm thời có thể thay đổi khi các chương trình con bị gọi.

Mnemonic	Number	Mnemonic	Number	Mnemonic	Number
\$zero	\$0		\$t3	\$11		\$s6	\$22
\$at	\$1		\$t4	\$12		\$s7	\$23
\$v0	\$2		\$t5	\$13		\$t8	\$24
\$v1	\$3		\$t6	\$14		\$t9	\$25
\$a0	\$4		\$t7	\$15		\$k0	\$26
\$a1	\$5		\$s0	\$16		\$k1	\$27
\$a2	\$6		\$s1	\$17		\$gp	\$28
\$a3	\$7		\$s2	\$18		\$sp	\$29
\$t0	\$8		\$s3	\$19		\$fp	\$30
\$t1	\$9		\$s4	\$20		\$ra	\$31
\$t2	\$10		\$s5	\$21			

Thanh ghi

\$s0-\$s7 (\$16-\$23):

- Được sử dụng để lưu các giá trị nhớ.
- Các giá trị của những thanh ghi này được duy trì qua các lời gọi hàm con.

Mnemonic	Number	Mnemonic	Number	Mnemonic	Number
\$zero	\$0		\$t3	\$11		\$s6	\$22
\$at	\$1		\$t4	\$12		\$s7	\$23
\$v0	\$2		\$t5	\$13		\$t8	\$24
\$v1	\$3		\$t6	\$14		\$t9	\$25
\$a0	\$4		\$t7	\$15		\$k0	\$26
\$a1	\$5		\$s0	\$16		\$k1	\$27
\$a2	\$6		\$s1	\$17		\$gp	\$28
\$a3	\$7		\$s2	\$18		\$sp	\$29
\$t0	\$8		\$s3	\$19		\$fp	\$30
\$t1	\$9		\$s4	\$20		\$ra	\$31
\$t2	\$10		\$s5	\$21			

Thanh ghi

\$k0-\$k1 (\$26-\$27)

- Được sử dụng bởi hệ điều hành
- Không khả dụng đối với người lập trình.

Mnemonic	Number	Mnemonic	Number	Mnemonic	Number
\$zero	\$0		\$t3	\$11		\$s6	\$22
\$at	\$1		\$t4	\$12		\$s7	\$23
\$v0	\$2		\$t5	\$13		\$t8	\$24
\$v1	\$3		\$t6	\$14		\$t9	\$25
\$a0	\$4		\$t7	\$15		\$k0	\$26
\$a1	\$5		\$s0	\$16		\$k1	\$27
\$a2	\$6		\$s1	\$17		\$gp	\$28
\$a3	\$7		\$s2	\$18		\$sp	\$29
\$t0	\$8		\$s3	\$19		\$fp	\$30
\$t1	\$9		\$s4	\$20		\$ra	\$31
\$t2	\$10		\$s5	\$21			

Thanh ghi

\$gp (\$28)

- Con trỏ tới bộ nhớ toàn cục, sử dụng trong cấp phát bộ nhớ động (heap allocation)

Mnemonic	Number	Mnemonic	Number	Mnemonic	Number
\$zero	\$0		\$t3	\$11		\$s6	\$22
\$at	\$1		\$t4	\$12		\$s7	\$23
\$v0	\$2		\$t5	\$13		\$t8	\$24
\$v1	\$3		\$t6	\$14		\$t9	\$25
\$a0	\$4		\$t7	\$15		\$k0	\$26
\$a1	\$5		\$s0	\$16		\$k1	\$27
\$a2	\$6		\$s1	\$17		\$gp	\$28
\$a3	\$7		\$s2	\$18		\$sp	\$29
\$t0	\$8		\$s3	\$19		\$fp	\$30
\$t1	\$9		\$s4	\$20		\$ra	\$31
\$t2	\$10		\$s5	\$21			

32 thanh ghi

\$sp (\$29):

- Con trỏ ngăn xếp
- Được sử dụng để trỏ đến điểm bắt đầu của dữ liệu trong ngăn xếp

Mnemonic	Number	Mnemonic	Number	Mnemonic	Number
\$zero	\$0		\$t3	\$11		\$s6	\$22
\$at	\$1		\$t4	\$12		\$s7	\$23
\$v0	\$2		\$t5	\$13		\$t8	\$24
\$v1	\$3		\$t6	\$14		\$t9	\$25
\$a0	\$4		\$t7	\$15		\$k0	\$26
\$a1	\$5		\$s0	\$16		\$k1	\$27
\$a2	\$6		\$s1	\$17		\$gp	\$28
\$a3	\$7		\$s2	\$18		\$sp	\$29
\$t0	\$8		\$s3	\$19		\$fp	\$30
\$t1	\$9		\$s4	\$20		\$ra	\$31
\$t2	\$10		\$s5	\$21			

32 thanh ghi

\$fp (\$30)

- Con trỏ khung
- Sử dụng với \$sp để duy trì thông tin về ngăn xếp.

Mnemonic	Number	Mnemonic	Number	Mnemonic	Number
\$zero	\$0		\$t3	\$11		\$s6	\$22
\$at	\$1		\$t4	\$12		\$s7	\$23
\$v0	\$2		\$t5	\$13		\$t8	\$24
\$v1	\$3		\$t6	\$14		\$t9	\$25
\$a0	\$4		\$t7	\$15		\$k0	\$26
\$a1	\$5		\$s0	\$16		\$k1	\$27
\$a2	\$6		\$s1	\$17		\$gp	\$28
\$a3	\$7		\$s2	\$18		\$sp	\$29
\$t0	\$8		\$s3	\$19		\$fp	\$30
\$t1	\$9		\$s4	\$20		\$ra	\$31
\$t2	\$10		\$s5	\$21			

32 thanh ghi

\$ra (\$31):

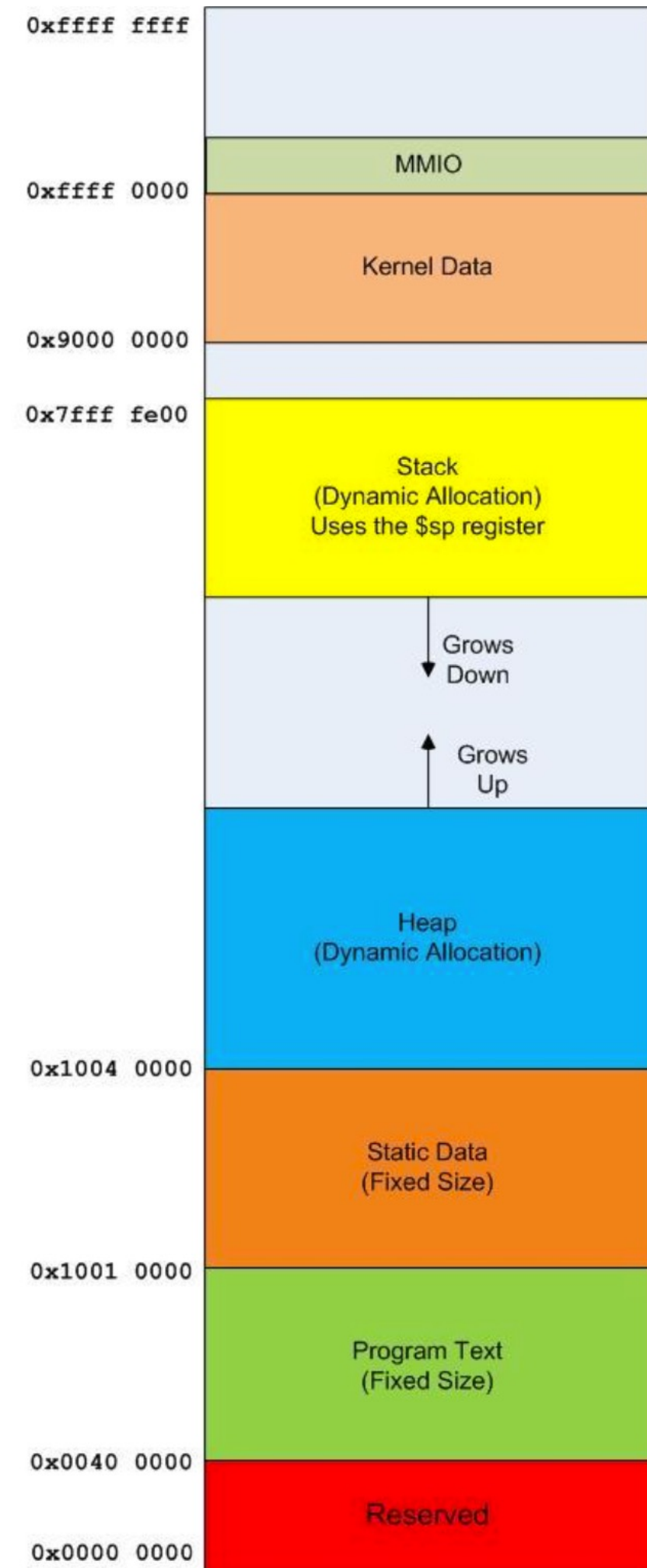
- Địa chỉ trả về
- Con trỏ trỏ tới địa chỉ sử dụng khi quay về từ 1 chương trình con.

Mnemonic	Number	Mnemonic	Number	Mnemonic	Number
\$zero	\$0		\$t3	\$11		\$s6	\$22
\$at	\$1		\$t4	\$12		\$s7	\$23
\$v0	\$2		\$t5	\$13		\$t8	\$24
\$v1	\$3		\$t6	\$14		\$t9	\$25
\$a0	\$4		\$t7	\$15		\$k0	\$26
\$a1	\$5		\$s0	\$16		\$k1	\$27
\$a2	\$6		\$s1	\$17		\$gp	\$28
\$a3	\$7		\$s2	\$18		\$sp	\$29
\$t0	\$8		\$s3	\$19		\$fp	\$30
\$t1	\$9		\$s4	\$20		\$ra	\$31
\$t2	\$10		\$s5	\$21			

Bộ nhớ

- Mô hình địa chỉ 32 bit phẳng
- Có thể đánh (tìm) địa chỉ cho 4GB dữ liệu
- Bắt đầu từ địa chỉ 0x00000000 đến 0xffffffff
- Không phải tất cả bộ nhớ đều khả dụng với người lập trình

Memory Address

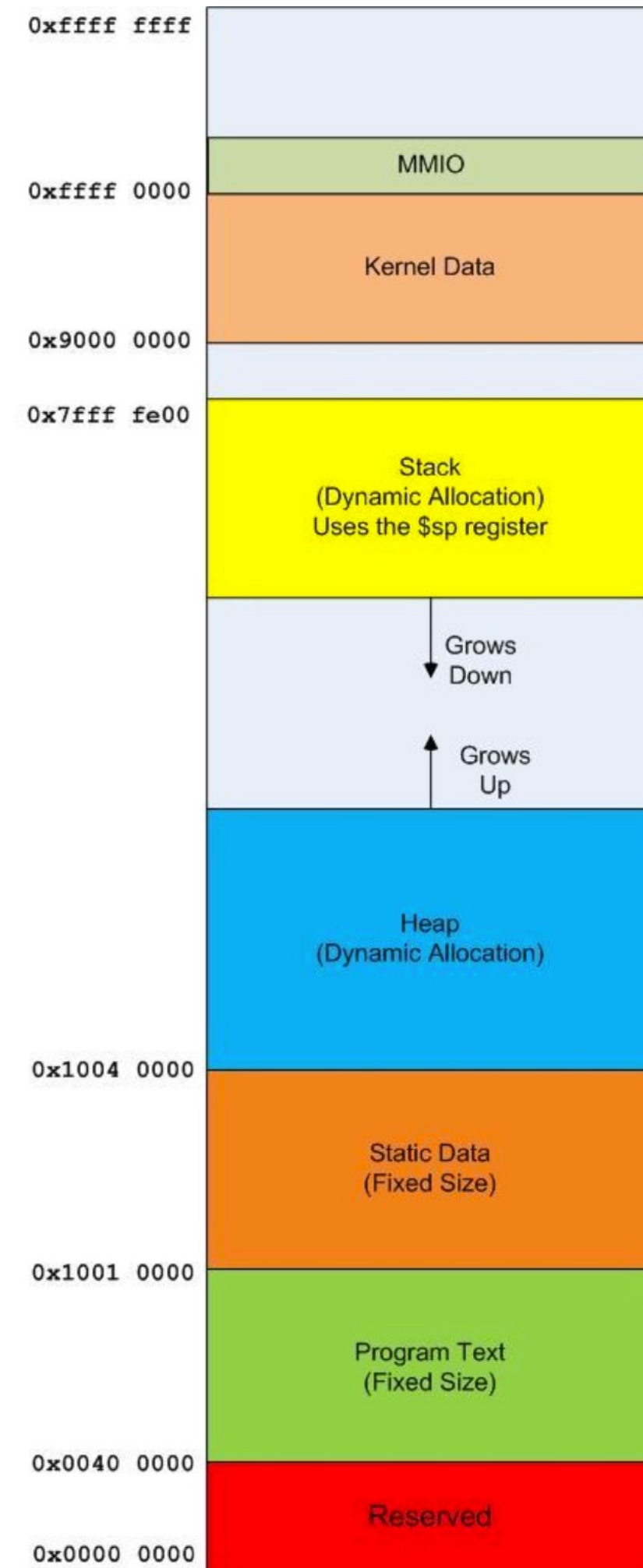


Các loại bộ nhớ

Reversed: Bộ nhớ Dự phòng

- Bộ nhớ được dành cho nền tảng MIPS.
- Bộ nhớ ở những địa chỉ này không khả dụng đối với chương trình.

Memory
Addresses



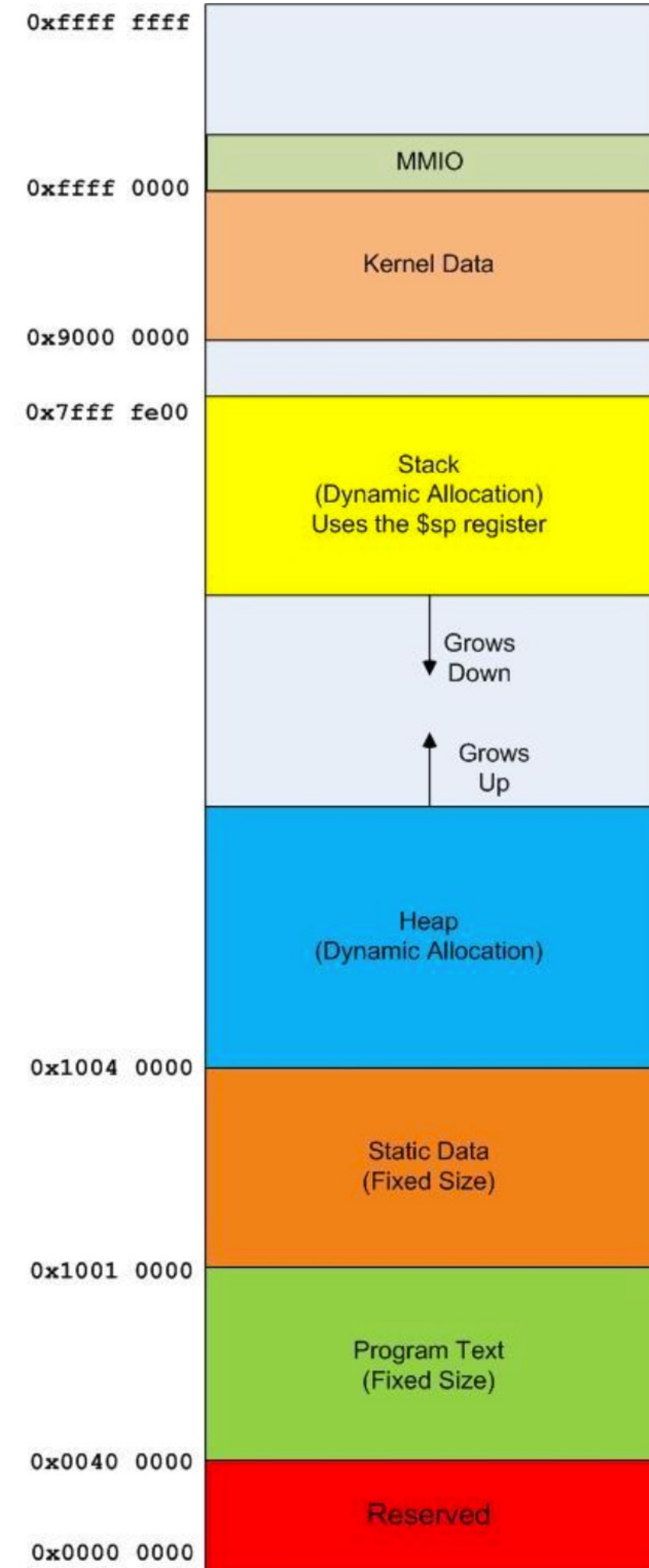
Các loại bộ nhớ

Program text: (dải địa chỉ từ 0x0040 0000 - 0x1000 00000)

- Lưu trữ biểu diễn mã máy của một chương trình.
- Mỗi lệnh được lưu như là 1 từ (word 32 bit hay 4 byte) trong bộ nhớ.
- Tất cả các lệnh đều có giới hạn từ, là bội số của 4 (0x0040 0000, 0x0040 0004, 0x0040 0080, 0x0040 00B0...)

M
e
m
o
r
y

A
d
d
r
e
s
s

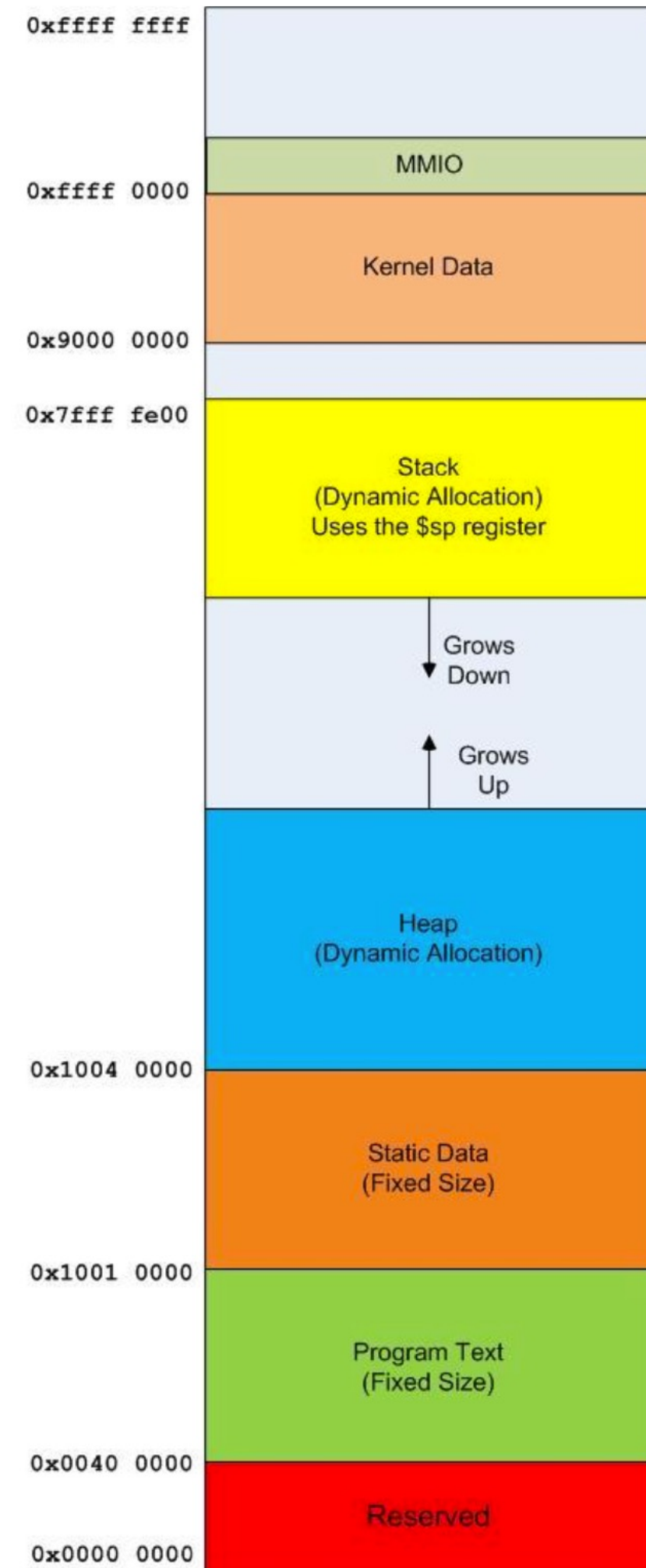


Các loại bộ nhớ

Static data: dữ liệu tĩnh (dải địa chỉ từ 0x1001 0000 - 0x1004 0000)

- Dữ liệu tĩnh đến từ data segment (phân đoạn dữ liệu) của chương trình.
- Kích thước các phần tử trong vùng này được gán khi chương trình được tạo ra (hợp dịch và liên kết) và không thể thay đổi trong suốt quá trình thực thi của chương trình.

Memory Address

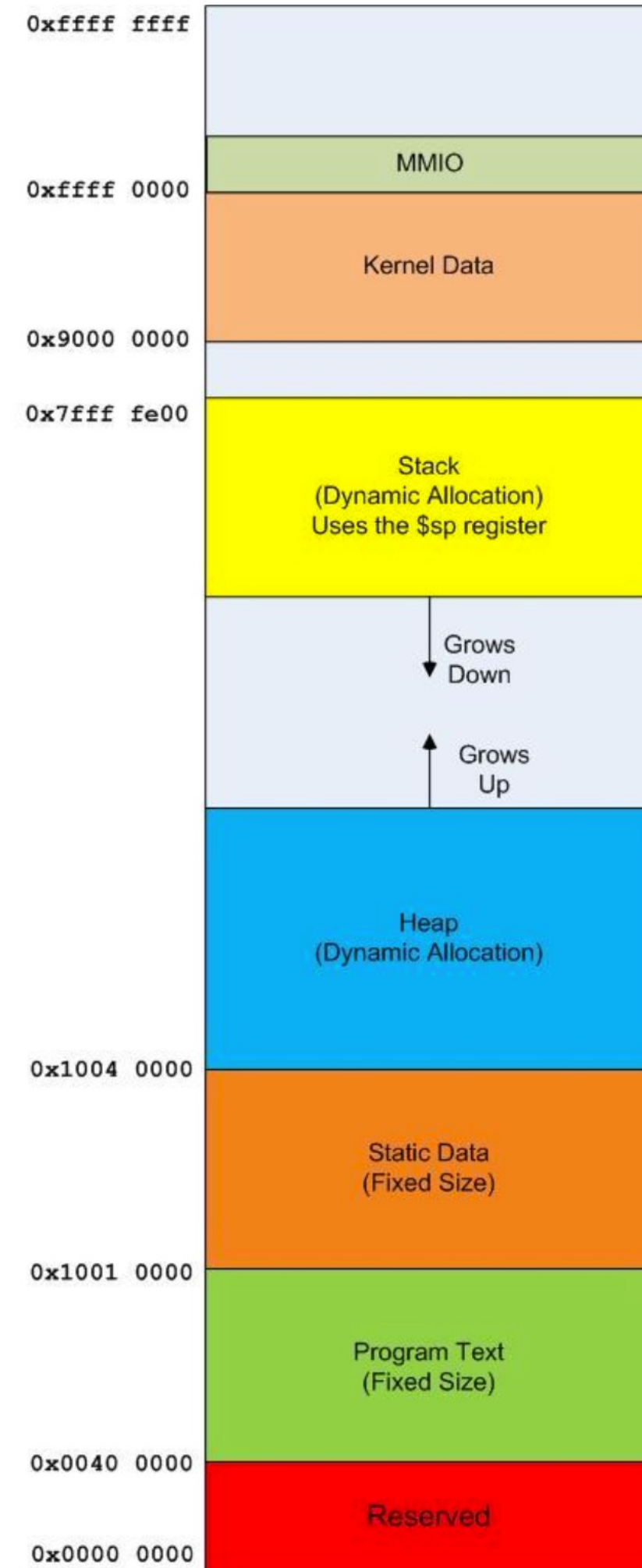


Các loại bộ nhớ

Heap – (địa chỉ từ 0x1004 0000 – cho đến khi tới stack, đi lên)

- Là dữ liệu động được cấp phát nếu cần thiết trong quá trình chạy chương trình.
- Cách bộ nhớ này được cấp phát và khai báo tùy thuộc vào ngôn ngữ cụ thể.
- Dữ liệu trong heap luôn luôn là dữ liệu toàn thể (toàn cục).

Memory
Addresses

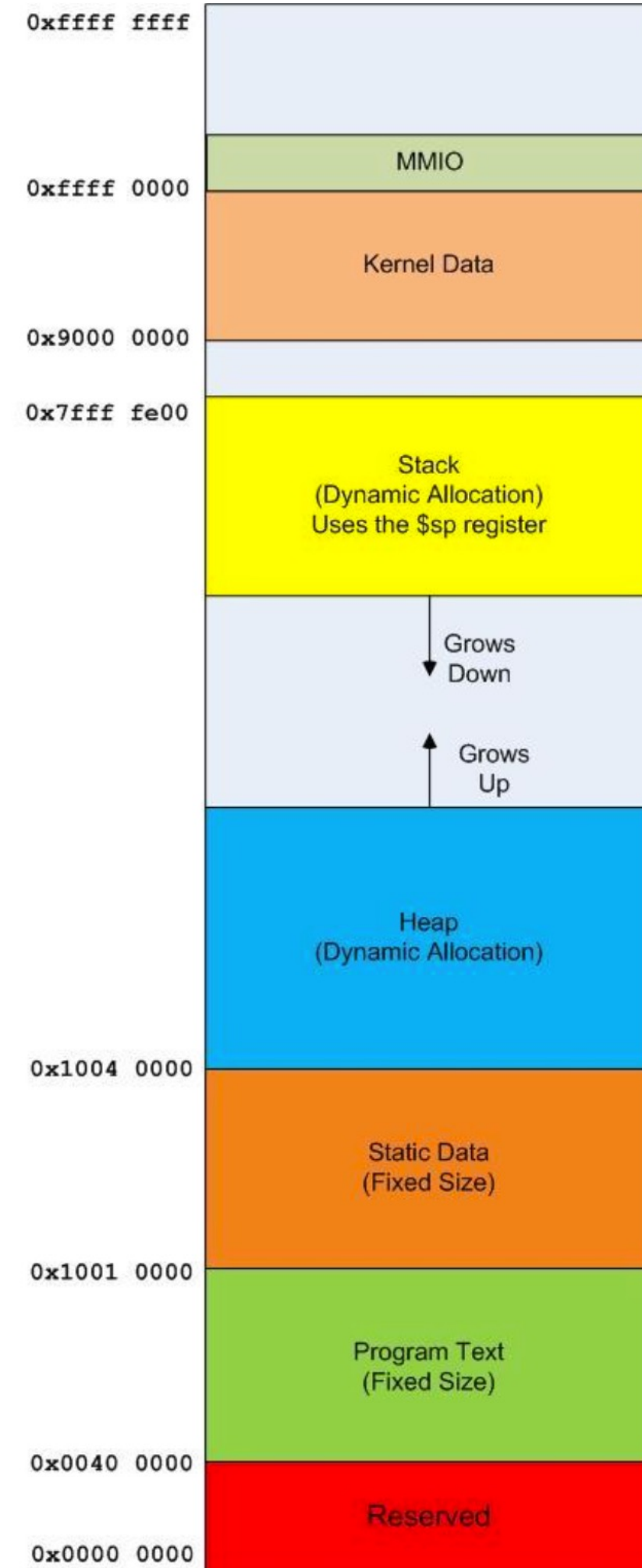


Các loại bộ nhớ

Stack - (địa chỉ từ 0x7fff fe00 – cho đến khi tới heap, đi xuống).

- Ngăn xếp chương trình là dữ liệu động được cấp phát cho các chương trình con theo toán tử push và pop.
- Tất cả các phương thức biến cục bộ được lưu trữ tại đây. Bởi vì tính tự nhiên của toán tử pop và push, kích thước của bản ghi ngăn xếp khi được tạo ra cần phải được biết trước khi chương trình được hợp dịch.

Memory
Addresses

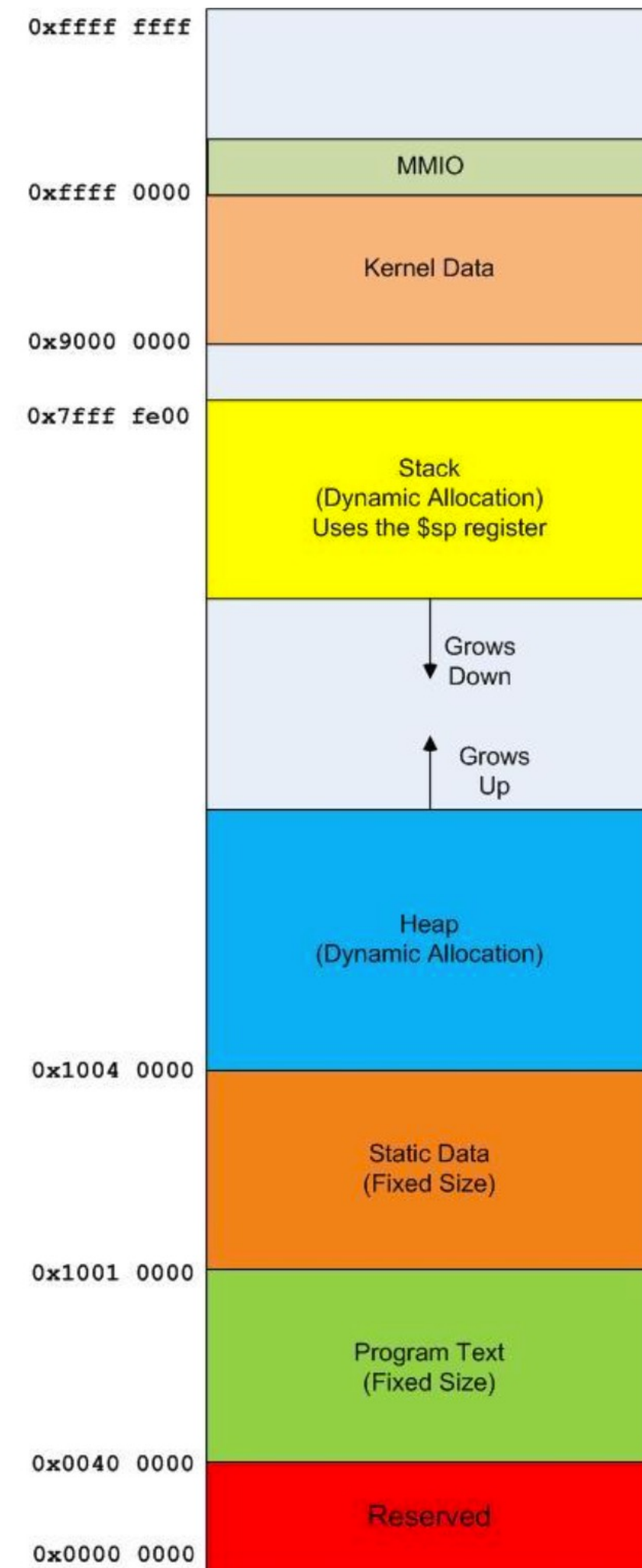


Các loại bộ nhớ

Kernel: (dải địa chỉ từ 0x9000 0000 - 0xffff 0000)

- Được sử dụng bởi hệ điều hành.
- Người sử dụng không được quyền truy cập.

Memory
Addresses

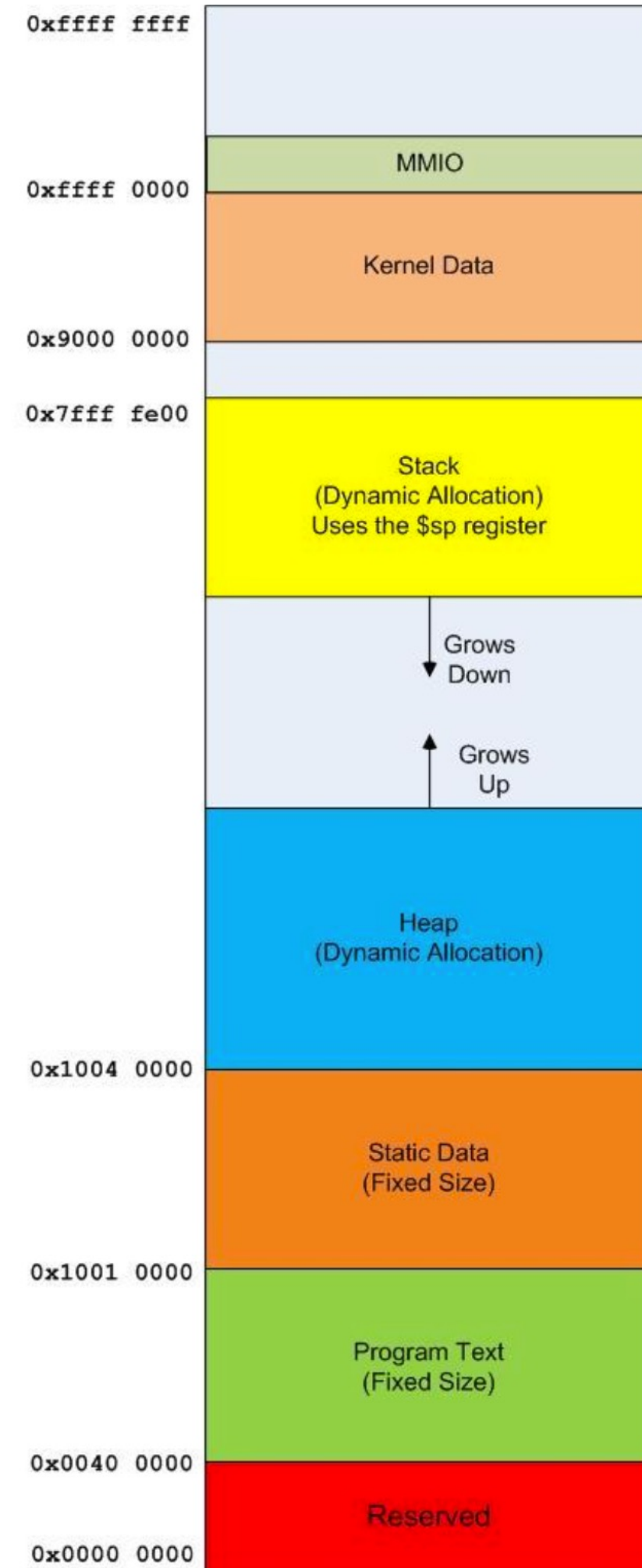


Các loại bộ nhớ

MMIO: (dải địa chỉ từ 0xffff 0000 - 0xffff 0010):

- Là bộ nhớ ánh xạ ra/vào, được sử dụng cho bất kỳ loại dữ liệu bên ngoài nào không ở trong bộ nhớ như là màn hình, ổ đĩa, console...

Memory
Addresses



Chương trình đầu tiên

- Cách chú thích một chương trình MIPS
- Thanh ghi và bộ nhớ trong máy tính MIPS
- Các chỉ thị hợp ngữ như **.text**, **.data**, **.asciiz** và **.word**
- Nhãn trong MIPS
- Các toán tử hợp ngữ MIPS như **li**, **la**, **lw** và **move**
- Các dịch vụ hệ thống để giao tiếp với console người dùng, cụ thể là các dịch vụ 1, 4, 5 và 8
- Sự khác nhau giữa giá trị tham chiếu và tham trị của dữ liệu

Program 2.1 Hello World

```
1  # Program File: Program 2-1.asm
2  # Author: NTTNga
3  # Purpose: First program, Hello World
4  .data                                # Define the program data.
5  greeting: .asciiz "Hello World" #The string to print.
6
7  .text                                # Define the program instructions.
8  main:                                # Label to define the main program.
9      li $v0,4                          # Load 4 into $v0 to indicate a print string.
10     la $a0,greeting                  # Load the address of the greeting into $a0.
11     syscall                          # Print greeting. The print is indicated by
12                                         # $v0 having a value of 4, and the string to
13                                         # print is stored at the address in $a0.
14     li $v0,10                        # Load a 10 (halt) into $v0.
15     syscall                          # The program ends.
```

- Mã trình biên dịch MIPS có thể được viết thụt lề.
- Dòng trắng được tự động bỏ qua.

Program 2.1 Hello World

```
1  # Program File: Program 2-1.asm
2  # Author: NTTNga
3  # Purpose: First program, Hello World
4  .data                                # Define the program data.
5  greeting: .asciiz "Hello World" #The string to print.
6
7  .text                                # Define the program instructions.
8  main:                                # Label to define the main program.
9      li $v0,4                          # Load 4 into $v0 to indicate a print string.
10     la $a0,greeting                  # Load the address of the greeting into $a0.
11     syscall                          # Print greeting. The print is indicated by
12                                     # $v0 having a value of 4, and the string to
13                                     # print is stored at the address in $a0.
14     li $v0,10                        # Load a 10 (halt) into $v0.
15     syscall                          # The program ends.
```

- Câu lệnh phải được viết trên cùng 1 dòng.

Program 2.1 Hello World

```
1 # Program File: Program 2-1.asm
2 # Author: NTTNga
3 # Purpose: First program, Hello World
4 .data                                # Define the program data.
5 greeting: .asciiz "Hello World"    # The string to print.
6
7 .text                                # Define the program instructions.
8 main:                               # Label to define the main program.
9     li $v0,4                        # Load 4 into $v0 to indicate a print string.
10    la $a0,greeting                 # Load the address of the greeting into $a0.
11    syscall                         # Print greeting. The print is indicated by
12                                     # $v0 having a value of 4, and the string to
13                                     # print is stored at the address in $a0.
14    li $v0,10                       # Load a 10 (halt) into $v0.
15    syscall                         # The program ends.
```

- Dấu # có nghĩa là bất kỳ ký tự nào từ dấu # cho đến hết dòng là chú thích và có thể được bỏ qua.

Program 2.1 Hello World

```
1  # Program File: Program 2-1.asm
2  # Author: NTTNga
3  # Purpose: First program, Hello World
4  .data                                # Define the program data.
5  greeting: .asciiz "Hello World"    #The string to print.
6
7  .text                                # Define the program instructions.
8  main:                                # Label to define the main program.
9      li $v0,4                          # Load 4 into $v0 to indicate a print string.
10     la $a0,greeting                  # Load the address of the greeting into $a0.
11     syscall                          # Print greeting. The print is indicated by
12                                     # $v0 having a value of 4, and the string to
13                                     # print is stored at the address in $a0.
14     li $v0,10                        # Load a 10 (halt) into $v0.
15     syscall                          # The program ends.
```

- Chuỗi được biểu thị bằng nội dung nằm trong dấu nháy kép “”.

Program 2.1 Hello World

```
1  # Program File: Program 2-1.asm
2  # Author: NTTNga
3  # Purpose: First program, Hello World
4  .data                                # Define the program data.
5  greeting: .asciiz "Hello World" #The string to print.
6
7  .text                                # Define the program instructions.
8  main:                                # Label to define the main program.
9      li $v0,4                          # Load 4 into $v0 to indicate a print string.
10     la $a0,greeting                   # Load the address of the greeting into $a0.
11     syscall                           # Print greeting. The print is indicated by
12                                         # $v0 having a value of 4, and the string to
13                                         # print is stored at the address in $a0.
14     li $v0,10                         # Load a 10 (halt) into $v0.
15     syscall                           # The program ends.
```

- Ghi chú lại các chú thích ở đầu mỗi tệp tin.
- Các ghi chú này được gọi là phần mở đầu của chương trình.

Program 2.1 Hello World

```
1  # Program File: Program 2-1.asm
2  # Author: NTTNga
3  # Purpose: First program, Hello World
4  .data                                # Define the program data.
5  greeting: .asciiz "Hello World" #The string to print.
6
7  .text                                # Define the program instructions.
8  main:                                # Label to define the main program.
9      li $v0,4                          # Load 4 into $v0 to indicate a print string.
10     la $a0,greeting                  # Load the address of the greeting into $a0.
11     syscall                          # Print greeting. The print is indicated by
12                                     # $v0 having a value of 4, and the string to
13                                     # print is stored at the address in $a0.
14     li $v0,10                        # Load a 10 (halt) into $v0.
15     syscall                          # The program ends.
```

- Các chương trình hợp ngữ không được **biên dịch**, chúng được **hợp dịch**.

Program 2.1 Hello World

```
1  # Program File: Program 2-1.asm
2  # Author: NTTNga
3  # Purpose: First program, Hello World
4  .data                                # Define the program data.
5  greeting: .asciiz "Hello World" #The string to print.
6
7  .text                                # Define the program instructions.
8  main:                                # Label to define the main program.
9      li $v0,4                          # Load 4 into $v0 to indicate a print string.
10     la $a0,greeting                   # Load the address of the greeting into $a0.
11     syscall                           # Print greeting. The print is indicated by
12                                         # $v0 having a value of 4, and the string to
13                                         # print is stored at the address in $a0.
14     li $v0,10                         # Load a 10 (halt) into $v0.
15     syscall                           # The program ends.
```

- Dấu “.” trước một chuỗi văn bản có nghĩa là mã (chuỗi) tiếp theo là một chỉ thị của bộ hợp ngữ.

Program 2.1 Hello World

```
1  # Program File: Program 2-1.asm
2  # Author: NTTNga
3  # Purpose: First program, Hello World
4  .data                                # Define the program data.
5  greeting: .asciiz "Hello World" #The string to print.
6
7  .text                                # Define the program instructions.
8  main:                               # Label to define the main program.
9      li $v0,4                        # Load 4 into $v0 to indicate a print string.
10     la $a0,greeting                 # Load the address of the greeting into $a0.
11     syscall                         # Print greeting. The print is indicated by
12                                     # $v0 having a value of 4, and the string to
13                                     # print is stored at the address in $a0.
14     li $v0,10                       # Load a 10 (halt) into $v0.
15     syscall                         # The program ends.
```

- Chỉ thị **.text** có nghĩa là các lệnh tiếp theo thuộc về văn bản chương trình (gọi tắt là chương trình), được tập hợp thành một chương trình và lưu trữ trong vùng văn bản của bộ nhớ.

Program 2.1 Hello World

```
1  # Program File: Program 2-1.asm
2  # Author: NTTNga
3  # Purpose: First program, Hello World
4  .data                                # Define the program data.
5  greeting: .asciiz "Hello World" #The string to print.
6
7  .text                                # Define the program instructions.
8  main:                               # Label to define the main program.
9      li $v0,4                        # Load 4 into $v0 to indicate a print string.
10     la $a0,greeting                 # Load the address of the greeting into $a0.
11     syscall                         # Print greeting. The print is indicated by
12                                     # $v0 having a value of 4, and the string to
13                                     # print is stored at the address in $a0.
14     li $v0,10                       # Load a 10 (halt) into $v0.
15     syscall                         # The program ends.
```

- Chỉ thị **.data** có nghĩa là những gì tiếp theo là dữ liệu chương trình và được lưu trữ trong vùng dữ liệu tĩnh của bộ nhớ.

Program 2.1 Hello World

```
1  # Program File: Program 2-1.asm
2  # Author: NTTNga
3  # Purpose: First program, Hello World
4  .data                                # Define the program data.
5  greeting: .ascii "Hello World" #The string to print.
6
7  .text                                # Define the program instructions.
8  main:                                # Label to define the main program.
9      li $v0,4                          # Load 4 into $v0 to indicate a print string.
10     la $a0,greeting                  # Load the address of the greeting into $a0.
11     syscall                          # Print greeting. The print is indicated by
12                                     # $v0 having a value of 4, and the string to
13                                     # print is stored at the address in $a0.
14     li $v0,10                        # Load a 10 (halt) into $v0.
15     syscall                          # The program ends.
```

- Chỉ thị **.ascii** yêu cầu trình biên dịch dịch dữ liệu tiếp theo như là một chuỗi ký tự ASCII.

Program 2.1 Hello World

```
1  # Program File: Program 2-1.asm
2  # Author: NTTNga
3  # Purpose: First program, Hello World
4  .data                                # Define the program data.
5  greeting: .asciiz "Hello World" #The string to print.
6
7  .text                                # Define the program instructions.
8  main:                               # Label to define the main program.
9      li $v0,4                        # Load 4 into $v0 to indicate a print string.
10     la $a0,greeting                # Load the address of the greeting into $a0.
11     syscall                        # Print greeting. The print is indicated by
12                                     # $v0 having a value of 4, and the string to
13                                     # print is stored at the address in $a0.
14     li $v0,10                      # Load a 10 (halt) into $v0.
15     syscall                        # The program ends.
```

- Chuỗi văn bản nào theo sau dấu “:” được gọi là nhãn.
- Một nhãn chỉ là một điểm đánh dấu trong chương trình để có thể được sử dụng trong các câu lệnh khác.

Program 2.1 Hello World

```
1  # Program File: Program 2-1.asm
2  # Author: NTTNga
3  # Purpose: First program, Hello World
4  .data                                # Define the program data.
5  greeting: .asciiz "Hello World" #The string to print.
6
7  .text                                # Define the program instructions.
8  main:                                # Label to define the main program.
9      li $v0,4                          # Load 4 into $v0 to indicate a print string.
10     la $a0,greeting                  # Load the address of the greeting into $a0.
11     syscall                          # Print greeting. The print is indicated by
12                                     # $v0 having a value of 4, and the string to
13                                     # print is stored at the address in $a0.
14     li $v0,10                        # Load a 10 (halt) into $v0.
15     syscall                          # The program ends.
```

- Nhãn **main**: không cần phải có trong chương trình vì MARS giả thiết rằng chương trình bắt đầu từ dòng đầu tiên trong chương trình hợp ngữ. Tuy nhiên, sẽ tốt hơn nếu ta gán nhãn cho điểm bắt đầu vì phần lớn thời gian chạy sẽ tìm kiếm một tên biểu trưng toàn thể **main** như là nơi để bắt đầu thực thi chương trình.

Program 2.1 Hello World

```
1  # Program File: Program 2-1.asm
2  # Author: NTTNga
3  # Purpose: First program, Hello World
4  .data                                # Define the program data.
5  greeting: .asciiz "Hello World" #The string to print.
6
7  .text                                # Define the program instructions.
8  main:                                # Label to define the main program.
9      li $v0, 4                        # Load 4 into $v0 to indicate a print string.
10     la $a0, greeting                 # Load the address of the greeting into $a0.
11     syscall                          # Print greeting. The print is indicated by
12                                     # $v0 having a value of 4, and the string to
13                                     # print is stored at the address in $a0.
14     li $v0, 10                       # Load a 10 (halt) into $v0.
15     syscall                          # The program ends.
```

- Bất kỳ khi nào một hằng số có trong một lệnh, nó được gọi là giá trị tức thời (Immediate I).
- Giá trị tức thời phải có mặt trong bản thân câu lệnh.

Program 2.1 Hello World

```
1  # Program File: Program 2-1.asm
2  # Author: NTTNga
3  # Purpose: First program, Hello World
4  .data                                # Define the program data.
5  greeting: .asciiz "Hello World" #The string to print.
6
7  .text                                # Define the program instructions.
8  main:                                # Label to define the main program.
9      li $v0,4                          # Load 4 into $v0 to indicate a print string.
10     la $a0,greeting                  # Load the address of the greeting into $a0.
11     syscall                          # Print greeting. The print is indicated by
12                                     # $v0 having a value of 4, and the string to
13                                     # print is stored at the address in $a0.
14     li $v0,10                        # Load a 10 (halt) into $v0.
15     syscall                          # The program ends.
```

- Chỉ có các **lệnh** và **nhãn** có thể được định nghĩa trong một vùng văn bản (text segment).
- Chỉ có **dữ liệu** và **nhãn** có thể được định nghĩa trong một vùng dữ liệu (data segment).

Program 2.1 Hello World

```
1  # Program File: Program 2-1.asm
2  # Author: NTTNga
3  # Purpose: First program, Hello World
4  .data                                # Define the program data.
5  greeting: .asciiz "Hello World" #The string to print.
6
7  .text                                # Define the program instructions.
8  main:                                # Label to define the main program.
9      li $v0,4                          # Load 4 into $v0 to indicate a print string.
10     la $a0,greeting                  # Load the address of the greeting into $a0.
11     syscall                          # Print greeting. The print is indicated by
12                                     # $v0 having a value of 4, and the string to
13                                     # print is stored at the address in $a0.
14     li $v0,10                        # Load a 10 (halt) into $v0.
15     syscall                          # The program ends.
```

- Các toán tử là các chuỗi ký tự giống như **li**, **la**, và **syscall**.

Program 2.1 Hello World

```
1  # Program File: Program 2-1.asm
2  # Author: NTTNga
3  # Purpose: First program, Hello World
4  .data                                # Define the program data.
5  greeting: .asciiz "Hello World" #The string to print.
6
7  .text                                # Define the program instructions.
8  main:                                # Label to define the main program.
9      li $v0,4                          # Load 4 into $v0 to indicate a print string.
10     la $a0,greeting                  # Load the address of the greeting into $a0.
11     syscall                          # Print greeting. The print is indicated by
12                                     # $v0 having a value of 4, and the string to
13                                     # print is stored at the address in $a0.
14     li $v0,10                        # Load a 10 (halt) into $v0.
15     syscall                          # The program ends.
```

- Một lệnh bao gồm toán tử và đối số của nó.
- Vì vậy, **li** là một toán tử; **li \$v0, 4** là một lệnh.

Program 2.1 Hello World

```
1  # Program File: Program 2-1.asm
2  # Author: NTTNga
3  # Purpose: First program, Hello World
4  .data                                # Define the program data.
5  greeting: .asciiz "Hello World" #The string to print.
6
7  .text                                # Define the program instructions.
8  main:                                # Label to define the main program.
9      li $v0,4                          # Load 4 into $v0 to indicate a print string.
10     la $a0,greeting                  # Load the address of the greeting into $a0.
11     syscall                          # Print greeting. The print is indicated by
12                                     # $v0 having a value of 4, and the string to
13                                     # print is stored at the address in $a0.
14     li $v0,10                        # Load a 10 (halt) into $v0.
15     syscall                          # The program ends.
```

- Toán tử **syscall** được sử dụng để gọi các dịch vụ hệ thống.

Tra cứu trong HELP

The screenshot shows the MARS 4.5 Help window. The title bar is 'MARS 4.5 Help'. The menu bar includes 'MIPS', 'MARS', 'License', 'Bugs/Comments', 'Acknowledgements', and 'Instruction Set Song'. The 'MIPS' menu is open, showing a list of topics: 'Basic Instructions', 'Extended (pseudo) Instructions', 'Directives', 'Syscalls' (highlighted), 'Exceptions', and 'Macros'. Below the menu, there is a section titled 'Operand Key for Example Instructions' with a light green background. It lists: 'label, target' (any textual label), '\$t1, \$t2, \$t3' (any integer register), '\$f2, \$f4, \$f6' (even-numbered floating point register), and '\$f0, \$f1, \$f3' (any floating point register). Below this, the 'Syscalls' section is shown. It contains two example instructions: 'li \$v0, 1 # service 1 is print integer' and 'add \$a0, \$t0, \$zero # load desired value into argument register \$a0, using pseudo-op syscall'. Below the examples is a table titled 'Table of Available Services'. The table has four columns: 'Service', 'Code in \$v0', 'Arguments', and 'Result'. It lists eight services: print integer, print float, print double, print string, read integer, read float, read double, and read string. The 'read string' service has a note 'See note below table' in the Result column.

Operand Key for Example Instructions

label, target	any textual label
\$t1, \$t2, \$t3	any integer register
\$f2, \$f4, \$f6	even-numbered floating point register
\$f0, \$f1, \$f3	any floating point register

Basic Instructions **Extended (pseudo) Instructions** **Directives** **Syscalls** **Exceptions** **Macros**

```
li $v0, 1          # service 1 is print integer
add $a0, $t0, $zero # load desired value into argument register $a0, using pseudo-op syscall
```

Table of Available Services

Service	Code in \$v0	Arguments	Result
print integer	1	\$a0 = integer to print	
print float	2	\$f12 = float to print	
print double	3	\$f12 = double to print	
print string	4	\$a0 = address of null-terminated string to print	
read integer	5		\$v0 contains integer read
read float	6		\$f0 contains float read
read double	7		\$f0 contains double read
read string	8	\$a0 = address of input buffer \$a1 = maximum number of characters to read	See note below table

Close

Program 2.2 - Chương trình nhắc và đọc vào một số nguyên từ người dùng

```
1  # Program File: Program 2-2.asm
2  # Author: NTTNga
3  # Program to read an integer number from a user, and
4  # print that number back to the console.
5
6  .data
7  prompt: .ascii "Hay nhap vao mot so nguyen: "
8  output: .ascii "\nBan da nhap vao so: "
9
10 .text
11 main:
12     # Prompt for the integer to enter
13     li $v0, 4
14     la $a0, prompt
15     syscall
16
17     # Read the integer and save it in $s0
18     li $v0, 5
19     syscall
20     move $s0, $v0
21
22     # Output the text
23     li $v0, 4
24     la $a0, output
25     syscall
26
27     # Output the number
28     li $v0, 1
29     move $a0, $s0
30     syscall
31
32     # Exit the program
33     li $v0, 10
34     syscall
```

- Chú thích theo khối lệnh
- Nên chú thích theo mỗi khối lệnh (chức năng/cách hoạt động của khối lệnh)

Program 2.2 - Chương trình nhắc và đọc vào một số nguyên từ người dùng

```
1  # Program File: Program 2-2.asm
2  # Author: NTTNga
3  # Program to read an integer number from a user, and
4  # print that number back to the console.
5
6  .data
7  prompt: .asciiz "Hay nhap vao mot so nguyen: "
8  output: .asciiz "\nBan da nhap vao so: "
9
10 .text
11 main:
12     # Prompt for the integer to enter
13     li $v0,4
14     la $a0,prompt
15     syscall
16
17     # Read the integer and save it in $s0
18     li $v0,5
19     syscall
20     move $s0,$v0
21
22     # Output the text
23     li $v0,4
24     la $a0,output
25     syscall
26
27     # Output the number
28     li $v0,1
29     move $a0,$s0
30     syscall
31
32     # Exit the program
33     li $v0,10
34     syscall
```

- Toán tử **move** chuyển nội dung từ thanh ghi này sang thanh ghi khác.

Program 2.2 - Chương trình nhắc và đọc vào một số nguyên từ người dùng

```
1  # Program File: Program 2-2.asm
2  # Author: NTTNga
3  # Program to read an integer number from a user, and
4  # print that number back to the console.
5
6  .data
7  prompt: .asciiz "Hay nhap vao mot so nguyen: "
8  output: .asciiz "\nBan da nhap vao so: "
9
10 .text
11 main:
12     # Prompt for the integer to enter
13     li $v0,4
14     la $a0,prompt
15     syscall
16
17     # Read the integer and save it in $s0
18     li $v0,5
19     syscall
20     move $s0,$v0
21
22     # Output the text
23     li $v0,4
24     la $a0,output
25     syscall
26
27     # Output the number
28     li $v0,1
29     move $a0,$s0
30     syscall
31
32     # Exit the program
33     li $v0,10
34     syscall
```

- Dịch vụ **5** chờ đồng bộ để người dùng nhập vào một số nguyên trên console
- Khi số nguyên được nhập sẽ trả về giá trị trong thanh ghi **\$v0**.
- Kiểm tra xem giá trị nhập có phải là một giá trị nguyên không
- Đưa ra một ngoại lệ nếu không phải.

Program 2.2 - Chương trình nhắc và đọc vào một số nguyên từ người dùng

```
1  # Program File: Program 2-2.asm
2  # Author: NTTNga
3  # Program to read an integer number from a user, and
4  # print that number back to the console.
5
6  .data
7  prompt: .ascii "Hay nhap vao mot so nguyen: "
8  output: .ascii "\nBan da nhap vao so: "
9
10 .text
11 main:
12     # Prompt for the integer to enter
13     li $v0,4
14     la $a0,prompt
15     syscall
16
17     # Read the integer and save it in $s0
18     li $v0,5
19     syscall
20     move $s0,$v0
21
22     # Output the text
23     li $v0,4
24     la $a0,output
25     syscall
26
27     # Output the number
28     li $v0,1
29     move $a0,$s0
30     syscall
31
32     # Exit the program
33     li $v0,10
34     syscall
```

- Dịch vụ 1 in ra giá trị số nguyên trong thanh ghi **\$a0**
- Chú ý:
 - Với dịch vụ 4, chuỗi ký tự lưu tại địa chỉ trong thanh ghi **\$a0** được in ra.
 - Với dịch vụ 1, giá trị trong thanh ghi **\$a0** cũng được in ra.

Program 2.2 - Chương trình nhắc và đọc vào một số nguyên từ người dùng

```
1  # Program File: Program 2-2.asm
2  # Author: NTTNga
3  # Program to read an integer number from a user, and
4  # print that number back to the console.
5
6  .data
7  prompt: .ascii "Hãy nhập vào một số nguyên: "
8  output: .ascii "\nBạn đã nhập vào số: "
9
10 .text
11 main:
12     # Prompt for the integer to enter
13     li $v0,4
14     la $a0,prompt
15     syscall
16
17     # Read the integer and save it in $s0
18     li $v0,5
19     syscall
20     move $s0,$v0
21
22     # Output the text
23     li $v0,4
24     la $a0,output
25     syscall
26
27     # Output the number
28     li $v0,1
29     move $a0,$s0
30     syscall
31
32     # Exit the program
33     li $v0,10
34     syscall
```

- Ký tự “\n” được sử dụng trong chuỗi ký tự được đặt tên là **output**.
- Được gọi là ký tự dòng mới, làm cho chuỗi **output** bắt đầu trên một dòng mới.

Program 2.3 - Chương trình nhắc và đọc vào một chuỗi từ người dùng

```
1 # Program File: Program 2-3.asm
2 # Author: NTTNga
3 # Program to read a string from a user, and
4 # print that string back to the console.
5
6 .data
7 input: .space 81
8 inputSize: .word 80
9 prompt: .asciiz "Hay nhap vao mot chuoi: "
10 output: .asciiz "\nBan da nhap vao chuoi: "
11
12 .text
13 main:
14     # Prompt for the string to enter
15     li $v0,4
16     la $a0,prompt
17     syscall
18
19     # Read the string.
20     li $v0,8
21     la $a0,input
22     lw $a1,inputSize
23     syscall
24
25     # Output the text
26     li $v0,4
27     la $a0,output
28     syscall
29
30     # Output the number
31     li $v0,4
32     la $a0,input
33     syscall
34
35     # Exit the program
36     li $v0,10
37     syscall
```

- Chỉ thị **.space** phân bổ **n bytes** của bộ nhớ trong vùng dữ liệu của chương trình với **n=81** trong chương trình này.
- Khi kích thước của 1 ký tự là 1 byte, thì điều này tương đương với lưu 80 ký tự dữ liệu.
- Tại sao lại khai báo không gian vùng nhớ là 81?

Program 2.3 - Chương trình nhắc và đọc vào một chuỗi từ người dùng

```
1  # Program File: Program 2-3.asm
2  # Author: NTTNga
3  # Program to read a string from a user, and
4  # print that string back to the console.
5
6  .data
7  input: .space 81
8  inputSize: .word 80
9  prompt: .asciiz "Hay nhap vao mot chuoi: "
10 output: .asciiz "\nBan da nhap vao chuoi: "
11
12 .text
13 main:
14     # Prompt for the string to enter
15     li $v0,4
16     la $a0,prompt
17     syscall
18
19     # Read the string.
20     li $v0,8
21     la $a0,input
22     lw $a1,inputSize
23     syscall
24
25     # Output the text
26     li $v0,4
27     la $a0,output
28     syscall
29
30     # Output the number
31     li $v0,4
32     la $a0,input
33     syscall
34
35     # Exit the program
36     li $v0,10
37     syscall
```

- Chỉ thị **.word** phân bố 4 bytes không gian bộ nhớ trong vùng dữ liệu.
- Sau chỉ thị **.word** có thể là một số nguyên, và vùng không gian nhớ có độ lớn tương ứng với giá trị số nguyên đó sẽ được khởi tạo.
- Có thể lưu bất kỳ loại dữ liệu nào trong vùng nhớ này.

Program 2.3 - Chương trình nhắc và đọc vào một chuỗi từ người dùng

```
1  # Program File: Program 2-3.asm
2  # Author: NTTNga
3  # Program to read a string from a user, and
4  # print that string back to the console.
5
6  .data
7  input: .space 81
8  inputSize: .word 80
9  prompt: .asciiz "Hay nhap vao mot chuoai: "
10 output: .asciiz "\nBan da nhap vao chuoai: "
11
12 .text
13 main:
14     # Prompt for the string to enter
15     li $v0,4
16     la $a0,prompt
17     syscall
18
19     # Read the string.
20     li $v0,8
21     la $a0,input
22     lw $a1,inputSize
23     syscall
24
25     # Output the text
26     li $v0,4
27     la $a0,output
28     syscall
29
30     # Output the number
31     li $v0,4
32     la $a0,input
33     syscall
34
35     # Exit the program
36     li $v0,10
37     syscall
```

- Toán tử **la** nạp địa chỉ của nhãn vào trong một thanh ghi.
- Phương pháp này được gọi là tham chiếu đến dữ liệu
- Được thể hiện bởi

\$a0 <= label

có nghĩa là giá trị của nhãn (địa chỉ bộ nhớ) được nạp vào trong một thanh ghi.

Program 2.3 - Chương trình nhắc và đọc vào một chuỗi từ người dùng

```
1  # Program File: Program 2-3.asm
2  # Author: NTTNga
3  # Program to read a string from a user, and
4  # print that string back to the console.
5
6  .data
7  input: .space 81
8  inputSize: .word 80
9  prompt: .asciiz "Hay nhap vao mot chuoai: "
10 output: .asciiz "\nBan da nhap vao chuoai: "
11
12 .text
13 main:
14     # Prompt for the string to enter
15     li $v0,4
16     la $a0,prompt
17     syscall
18
19     # Read the string.
20     li $v0,8
21     la $a0,input
22     lw $a1,inputSize
23     syscall
24
25     # Output the text
26     li $v0,4
27     la $a0,output
28     syscall
29
30     # Output the number
31     li $v0,4
32     la $a0,input
33     syscall
34
35     # Exit the program
36     li $v0,10
37     syscall
```

- Toán tử **lw** nạp giá trị chứa trong nhãn vào thanh ghi.
- Việc nạp giá trị vào trong thanh ghi được thể hiện như sau:

\$a1 <= M[label]

có nghĩa giá trị tại nhãn được nạp vào trong thanh ghi **\$a1**.

Program 2.3 - Chương trình nhắc và đọc vào một chuỗi từ người dùng

Address	Value (+0)	Value (+4)
0x10010000	l l e H	\0 \0 \n o
0x10010020	\0 \0 \0 \0	\0 \0 \0 \0
0x10010040	\0 \0 \0 \0	\0 \0 \0 \0
0x10010060	o a v	t o m
0x10010080	h c o	: i o u
0x100100a0	\0 \0 \0 \0	\0 \0 \0 \0
0x100100c0	\0 \0 \0 \0	\0 \0 \0 \0

Address	Value (+0)	Value (+4)
0x10010000	0x6c6c6548	0x00000a6f
0x10010020	0x00000000	0x00000000
0x10010040	0x00000000	0x00000000
0x10010060	0x6f617620	0x746f6d20
0x10010080	0x6863206f	0x3a696f75
0x100100a0	0x00000000	0x00000000
0x100100c0	0x00000000	0x00000000


- Một chuỗi ký tự là một chuỗi tuần tự các ký tự ASCII được kết thúc bởi một giá trị rỗng (**null**).
- Chuỗi 5 ký tự cần 6 bytes để lưu trữ.

Program 2.3 - Chương trình nhắc và đọc vào một chuỗi từ người dùng

```
1  # Program File: Program 2-3.asm
2  # Author: NTTNga
3  # Program to read a string from a user, and
4  # print that string back to the console.
5
6  .data
7  input: .space 81
8  inputSize: .word 80
9  prompt: .asciiz "Hay nhap vao mot chuoai: "
10 output: .asciiz "\nBan da nhap vao chuoai: "
11
12 .text
13 main:
14     # Prompt for the string to enter
15     li $v0,4
16     la $a0,prompt
17     syscall
18
19     # Read the string.
20     li $v0,8
21     la $a0,input
22     lw $a1,inputSize
23     syscall
24
25     # Output the text
26     li $v0,4
27     la $a0,output
28     syscall
29
30     # Output the number
31     li $v0,4
32     la $a0,input
33     syscall
34
35     # Exit the program
36     li $v0,10
37     syscall
```


- Dịch vụ **syscall 8** đọc một chuỗi từ console.
- Có 2 tham số được truyền cho dịch vụ:
 - Bộ nhớ sử dụng để lưu chuỗi (lưu trong thanh ghi **\$a0**)
 - Kích thước lớn nhất của chuỗi đọc vào (lưu trong thanh ghi **\$a1**)

Program 2.3 - Chương trình nhắc và đọc vào một chuỗi từ người dùng



Address	Value (+0)	Value (+4)	Value (+8)
0x10010000	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x10010020	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x10010040	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x10010060	o a v	t o m	u h c

Trước và sau khi nạp vào chuỗi “**Hello**”



Address	Value (+0)	Value (+4)	Value (+8)
0x10010000	l l e H	\0 \0 \n o	\0 \0 \0 \0
0x10010020	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x10010040	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x10010060	o a v	t o m	u h c
0x10010080	h c o	: i o u	\0 \0 \0

Bài tập 2.1

Gõ chương trình sau vào công cụ MARS.

```
#Laboratory Exercise 2, Assignment 2
.text
    lui    $s0, 0x2110          #put upper half of pattern in $s0
    ori    $s0, $s0, 0x003d      #put lower half of pattern in $s0
```

Sau đó:

- Sử dụng công cụ gỡ rối, Debug, chạy từng lệnh và dừng lại,
- Ở mỗi lệnh, quan sát cửa sổ Register và chú ý
 - o Sự thay đổi giá trị của thanh ghi \$s0
 - o Sự thay đổi giá trị của thanh ghi \$pc
- Ở cửa sổ Data Segment, hãy click vào hộp combo để chuyển tới quan sát các byte trong vùng lệnh .text.
 - o Kiểm tra xem các byte đầu tiên ở vùng lệnh trùng với cột nào trong cửa sổ Text Segment.

Bài tập 2.2

Gõ chương trình sau vào công cụ MARS.

```
#Laboratory Exercise 2, Assignment 3
.text
    li    $s0,0x2110003d #pseudo instruction=2 basic instructions
    li    $s1,0x2        #but if the immediate value is small, one ins
```

Sau đó:

- Biên dịch và quan sát các lệnh mã máy trong cửa sổ Text Segment. Giải thích điều bất thường?

Bài tập 2.3

- Viết một chương trình nhắc người dùng nhập vào loại bánh mà họ thích.
- Sau đó chương trình sẽ in ra dòng thông báo “So you like _____ pie”, phần gạch chân sẽ được thay thế bằng tên loại bánh vừa được nhập vào.
- Tính năng gì của syscall service 4 làm cho từ dấu nhắc không thể hiển thị được dòng thông báo trên cùng một dòng?

Bài tập 2.4

- Hãy viết một chương trình in ra một số ngẫu nhiên từ 1...100.

Bài tập 2.5

- Write a program which sleeps for 4 seconds before exiting.

Kết thúc tuần 2