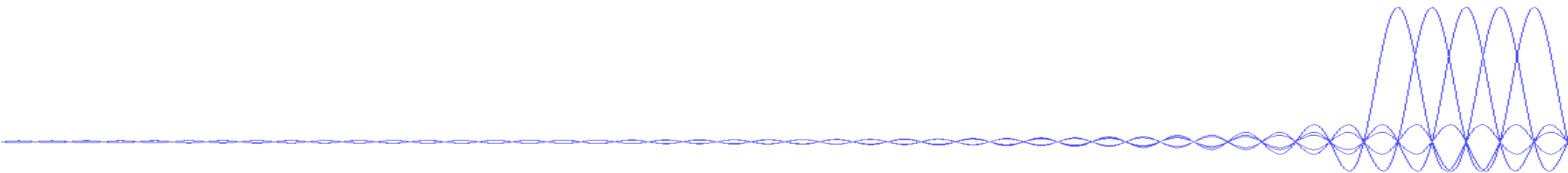
COMPUTER ENGINEERING

KIẾN TRÚC MÁY TÍNH

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

Tuần 5

KIẾN TRÚC BỘ LỆNH
(Tiếp theo)



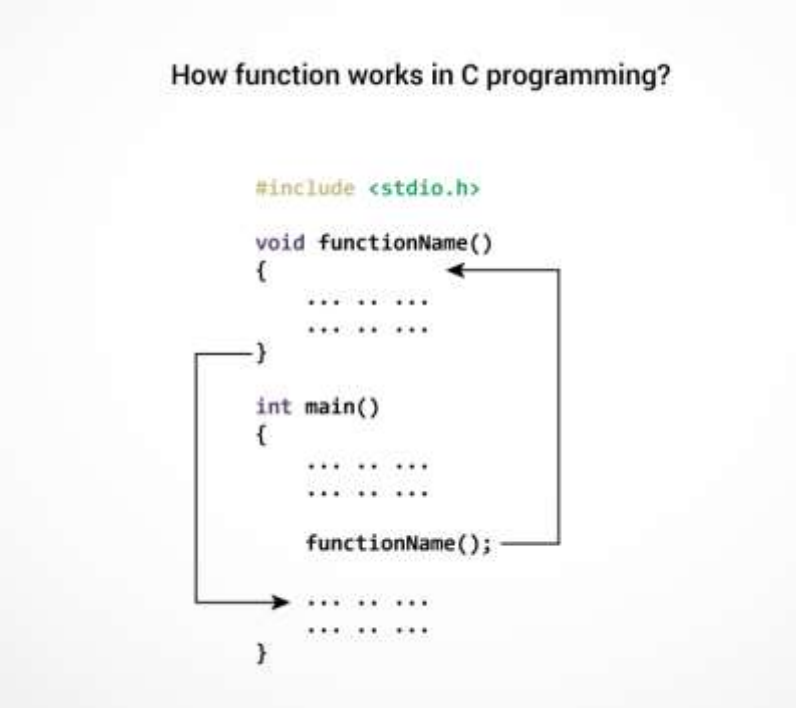
Mục tiêu:

1. Hiểu cách biểu diễn và cách thực thi các lệnh trong máy tính
2. Chuyển đổi lệnh ngôn ngữ cấp cao sang assembly và mã máy
3. Chuyển đổi lệnh mã máy sang ngôn ngữ cấp cao hơn
4. Biết cách lập trình bằng ngôn ngữ assembly cho MIPS

Slide được dịch và các hình được lấy từ sách tham khảo:

Computer Organization and Design: The Hardware/Software Interface, Patterson, D. A., and J. L. Hennessy, Morgan Kaufman, Revised Fourth Edition, 2011.

Thủ tục (Procedure) cho assembly MIPS




```
1  #include <iostream>
2  using namespace std;
3
4
5  int TongS(int n)
6  {
7      int Sum = 0;
8
9
10     for (int i = 1 ; i <= n ; i++)
11         Sum += i;
12
13     return Sum;
14 }
15
16
17 int main(int argc, const char * argv[]) {
18
19     int n;
20
21     cout << "Nhap gia tri n: " ;
22     cin >> n;
23
24     int S = TongS(n);
25     cout << "Tong cac so tu 1 den n: " << S;
26
27     return 0;
28 }
```

Hàm sẽ trả về dữ liệu có kiểu số nguyên int

Khai báo tên hàm và tham số đầu vào tương tự ví dụ 1

Kiểu dữ liệu của biến trả ra bên ngoài tương đồng với kiểu dữ liệu của hàm

Sử dụng lệnh return để trả một giá trị nào đó ra bên ngoài hàm. Lưu ý giá trị trả ra bên ngoài phải có kiểu dữ liệu tương ứng với kiểu dữ liệu của hàm.

Gọi sử dụng hàm. Do hàm có trả về dữ liệu kiểu int, nên sẽ cần khai báo một biến kiểu int để nhận kết quả trả về của hàm

In kết quả nhận được từ hàm ra màn hình

Các Thủ Tục Hỗ Trợ Trong Phần Cứng Máy Tính

❖ Một thủ tục (procedure) hay một hàm (function) là một công cụ mà lập trình viên sử dụng để xây dựng cấu trúc của những chương trình, với mục đích vừa làm cho các chương trình đó dễ hiểu hơn vừa làm cho mã nguồn của các chương trình này có thể được tái sử dụng.

✓Một chương trình có nhiều chức năng, mỗi chức năng sẽ được đưa vào một hàm, hoặc một thủ tục

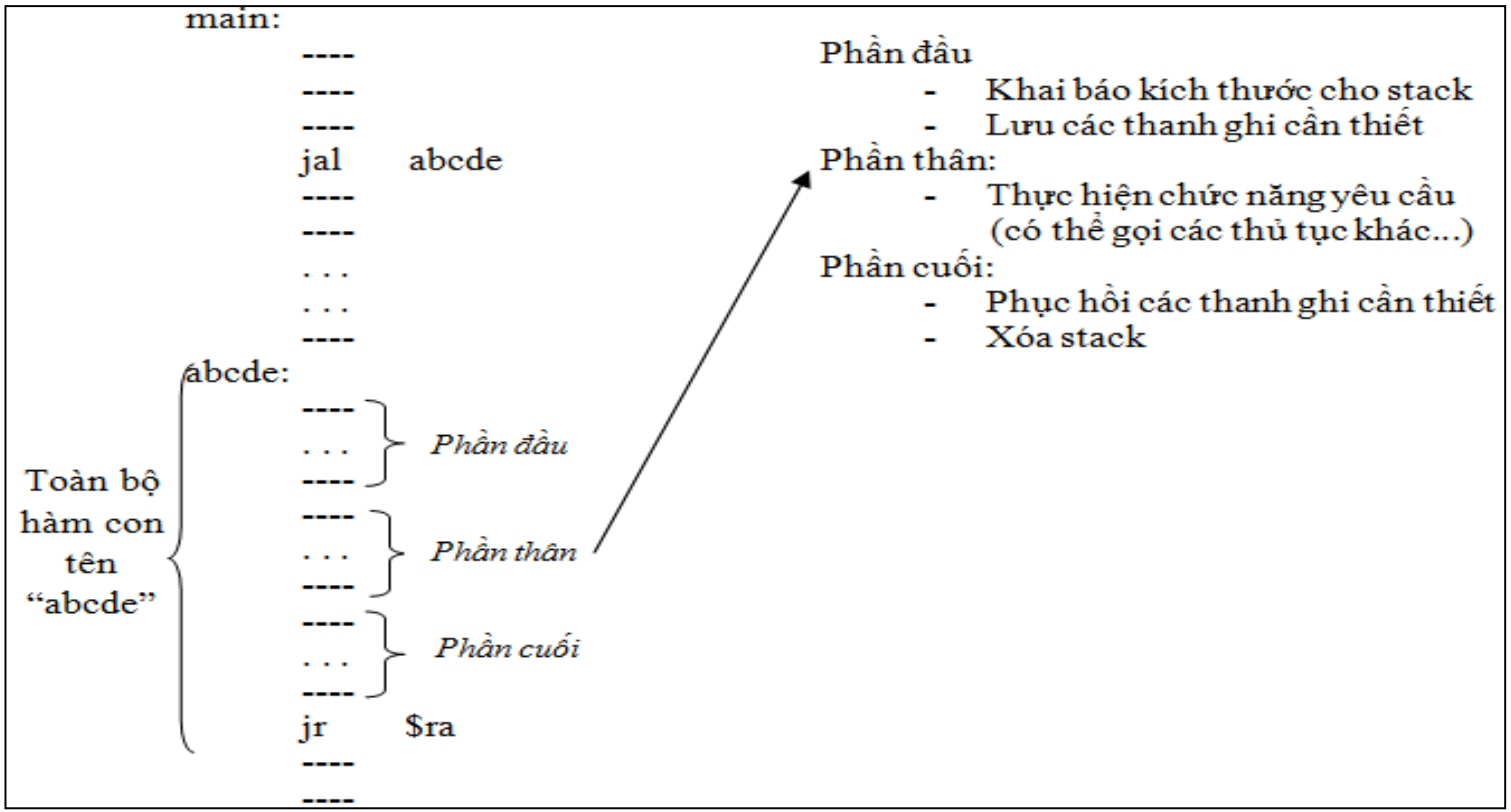
✓Các thủ tục hoặc hàm con này cho phép lập trình viên tại một thời điểm chỉ cần tập trung vào một phần của công việc, dễ dàng quản lý việc lập trình hơn

❖ Assembly cũng giống như các ngôn ngữ cấp cao, một chương trình với nhiều chức năng thì mỗi chức năng có thể đưa vào một thủ tục khác nhau.

Chú ý: Các thuật ngữ Routine/Procedure/Function có thể gặp trong một số môi trường khác nhau; trong assembly và phạm vi môn học này, tất cả đều được dịch là hàm hoặc thủ tục

Các Thủ Tục Hỗ Trợ Trong Phần Cứng Máy Tính

❖Cấu trúc một thủ tục/hàm con



Các Thủ Tục Hỗ Trợ Trong Phần Cứng Máy Tính

❖Các khái niệm và tên gọi:

▪**Program counter (PC):** Là thanh ghi chứa địa chỉ của lệnh kế tiếp được thực thi trong chương trình.

✓Thanh ghi PC còn được gọi là con trỏ PC hay con trỏ lệnh

✓Trong MIPS 32 bits, mỗi lệnh được lưu trong một word 4 bytes, do đó, để chỉ tới lệnh kế tiếp được thực thi trong chương trình, PC tăng lên 4 (ngoài trừ lệnh nhảy)

Các Thủ Tục Hỗ Trợ Trong Phần Cứng Máy Tính

❖ Để thực thi một thủ tục, chương trình phải tuân theo sáu bước sau:

1. *Đặt các tham số ở một nơi mà thủ tục có thể truy xuất được (truyền tham số)*
2. *Chuyển quyền điểu khiển cho thủ tục (PC = địa chỉ thủ tục)*
3. *Yêu cầu tài nguyên lưu trữ cần thiết cho thủ tục đó (phần đầu)*
4. *Thực hiện công việc (task) (phần thân)*
5. *Lưu kết quả ở một nơi mà chương trình có thể truy xuất được (phần cuối)*
6. *Trả điều khiển về vị trị mà thủ tục được gọi. Vì một thủ tục có thể được gọi từ nhiều vị trí trong một chương trình (PC = địa chỉ của lệnh ngay sau lệnh gọi thủ tục)*

Các Thủ Tục Hỗ Trợ Trong Phần Cứng Máy Tính

❖Cấu trúc một thủ tục/hàm con

Trong chương trình chính, khi thủ tục được gọi, con trỏ PC sẽ chuyển quyền điều khiển xuống vị trí của thủ tục; sau khi thủ tục được thực hiện xong, con trỏ PC sẽ chuyển về thực hiện lệnh ngay sau lệnh gọi thủ tục.

Việc này được thực hiện nhờ vào cặp lệnh:

jal ten_thu_tuc

jr $ra

(ten_thu_tuc là nhãn của lệnh đầu tiên trong thủ tục, cũng được xem là tên của thủ tục)

Các Thủ Tục Hỗ Trợ Trong Phần Cứng Máy Tính

❖Cấu trúc một thủ tục/hàm con

- Để gọi thủ tục, dùng lệnh ***jal*** *(Lệnh nhảy-và-liên kết: jump-and-link)*

✓Đầu tiên, địa chỉ của lệnh ngay sau lệnh *jal* phải được lưu lại để sau khi thủ tục thực hiện xong, con trỏ PC có thể chuyển về lại đây (thanh ghi dùng để lưu địa chỉ trả về là *$ra*); sau khi địa chỉ trả về đã được lưu lại, con trỏ PC chuyển đến địa chỉ của lệnh đầu tiên của thủ tục để thực hiện.

✓Vậy có hai công việc được thực hiện trong ***jal*** theo thứ tự:

```
jal <address>          #   $ra = PC + 4
                       #   PC = <address>
```

Các Thủ Tục Hỗ Trợ Trong Phần Cứng Máy Tính

❖Cấu trúc một thủ tục/hàm con

- Sau khi thủ tục thực hiện xong, để trả về lại chương trình chính, dùng lệnh ***jr*** *(jump regiser)*

✓Lệnh "jr $ra" đặt ở cuối thủ tục thực hiện việc lấy giá trị đang chứa trong thanh ghi $ra gán vào PC, giúp thanh ghi quay trở về thực hiện lệnh ngay sau lệnh gọi thủ tục này.

```
jr   $ra          #   PC = $ra
```

Các Thủ Tục Hỗ Trợ Trong Phần Cứng Máy Tính

❖Các khái niệm và tên gọi:

- Địa chỉ trả về (**return address**): là một liên kết tới vùng đang gọi cho phép một thủ tục trả về đúng địa chỉ; trong MIPS, nó được lưu trữ ở thanh ghi $ra.

Cụ thể: khi chương trình đang thực thi và một thủ tục được gọi, sau khi thủ tục thực thi xong, luồng thực thi lệnh phải quay về lại chương trình và thực hiện tiếp lệnh ngay phía sau thủ tục được gọi. Địa chỉ lệnh được thực thi sau khi thủ tục hoàn thành chính là địa chỉ trả về.

- **Caller:** Là chương trình gọi một thủ tục và cung cấp những giá trị tham số cần thiết.
- **Callee**: Là một thủ tục thực thi một chuỗi những lệnh được lưu trữ dựa trên những tham số được cung cấp bởi caller và sau đó trả điều khiều về cho caller.

Các Thủ Tục Hỗ Trợ Trong Phần Cứng Máy Tính

❖Cấu trúc một thủ tục/hàm con

Như vậy, mỗi hàm con/thủ tục được chia thành ba phần. Phần thân dùng để tính toán chức năng của thủ tục này, bắt buộc phải có. Phần đầu và phần cuối có thể có hoặc không, tùy từng yêu cầu của chương trình con.

❖Cấu trúc một thủ tục/hàm con

Ví dụ 1: Một thủ tục chỉ có thân hàm (không cần phần đầu và cuối)

Đoạn code sau thực hiện việc gọi hàm con (thủ tục) tên ABC

```
        add $t0, $t1, $t2
        jal ABC          ⟶  $ra = PC + 4
        or $t3, $t4, $s5        PC = ABC
        j EXIT
ABC:
        add $t0, $t3, $t5
        sub $t2, $t3, $t0
        jr $ra           ⟶  PC = $ra
EXIT:
```

Các Thủ Tục Hỗ Trợ Trong Phần Cứng Máy Tính

❖ Một số quy ước của MIPS với thanh ghi cần lưu ý:

- Thanh ghi $at ($1), $k0 ($26), $k1 ($27) được dành cho hệ điều hành và assembler; không nên sử dụng trong lúc lập trình thông thường.
- Thanh ghi $a0-$a3 ($4-$7) được sử dụng để truyền bốn tham số đến một *thủ tục* (nếu có hơn 4 tham số truyền vào, các tham số còn lại được lưu vào stack). Thanh ghi $v0-$v1 ($2-$3) được sử dụng để lưu giá trị trả về của hàm.
- Các thanh ghi $t0-$t9 ($9-$15, 24, 25) được sử dụng như các thanh ghi tạm. Các thanh ghi $s0-$s7 ($16-$23) được sử dụng như các thanh ghi lưu giá trị bền vững.

Trong MIPS, việc tính toán có thể cần một số thanh ghi trung gian, tạm thời, các thanh ghi $t nên được dùng cho mục đích này (nên thanh ghi nhóm $t thường gọi là thanh ghi tạm); còn kết quả cuối của phép toán nên lưu vào các thanh ghi $s (nên thanh ghi nhóm $s thường được gọi là nhóm thanh ghi lưu giá trị bền vững)

Các Thủ Tục Hỗ Trợ Trong Phần Cứng Máy Tính

❖ Một số quy ước của MIPS với thanh ghi cần lưu ý:

- Theo quy ước của MIPS, một thủ tục nếu sử dụng bất kỳ thanh ghi loại $s nào sẽ phải lưu lại giá trị của thanh ghi $s đó trước khi thực thi hàm. Và trước khi thoát ra khỏi hàm, các giá trị cũ của các thanh ghi $s này cũng phải được trả về lại. Các thanh ghi $s này được xem như biến cục bộ của hàm.

Thanh ghi loại $t được xem là các thanh ghi tạm, nên việc sử dụng thanh ghi loại $t trong thủ tục không cần lưu lại như các thanh ghi loại $s

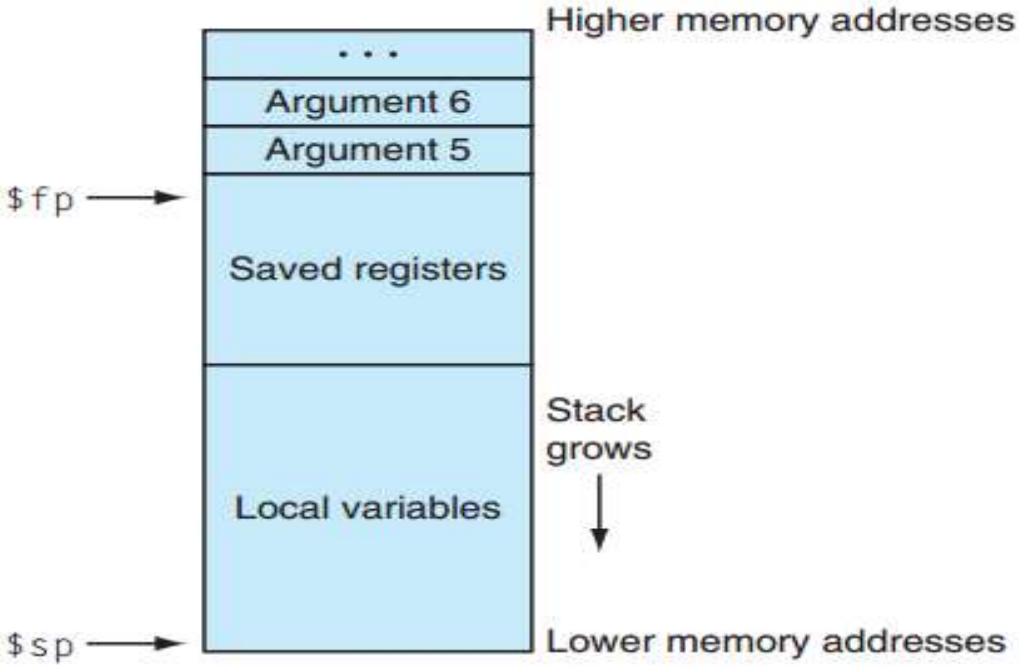
Vì thanh ghi là loại bộ nhớ có tốc độ truy xuất nhanh nhất, được dùng để lưu trữ dữ liệu trong một máy tính và số thanh ghi là hạn chế nên lập trình hợp ngữ phải hướng đến tận dụng thanh ghi một cách tối đa nhất.

Các Thủ Tục Hỗ Trợ Trong Phần Cứng Máy Tính

❖Các khái niệm và tên gọi:

- **Stack (ngăn xếp):** Là một cấu trúc dữ liệu trong bộ nhớ cho việc lưu dữ liệu theo hàng đợi dạng vào-sau ra-trước (last-in first-out)

Mỗi thủ tục luôn cần một vùng nhớ đi cùng với nó (vì mỗi thủ tục đều cần không gian để lưu lại các biến cục bộ của nó (là các thanh ghi $s được sử dụng trong thân hàm hoặc các biến cục bộ khác), hoặc để chứa các tham số input nếu số tham số lớn hơn 4 hoặc chứa giá trị trả về output nếu số giá trị trả về lớn hơn 2). Vùng nhớ này được quy ước hoạt động như một stack.

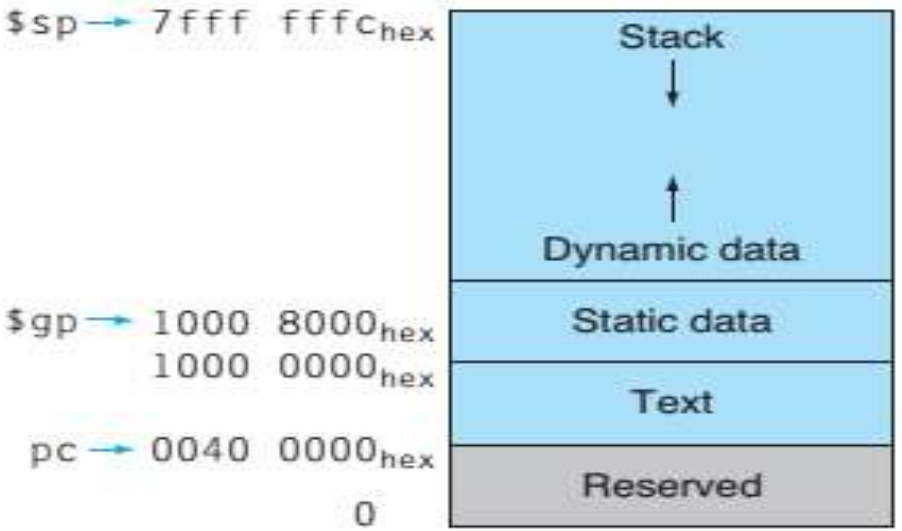


Hình ảnh ví dụ của một stack với các frame (stack này phục vụ cho thủ tục mà bốn tham số đầu vào đã được truyền thông qua thanh ghi, tham số thứ 5 và 6 được lưu trong stack như hình)

Các Thủ Tục Hỗ Trợ Trong Phần Cứng Máy Tính

Xem thêm về cách bố trí vùng **Stack, Heap, Data** và **Text Segment** trong bộ nhớ của một chương trình

- **Heap:** Là khu vực bộ nhớ cấp phát động, có thể được cấp phát thêm khi đạt đến giới hạn.
- **Text segment:** Đoạn mã chương trình.



Cấp phát bộ nhớ cho chương trình và dữ liệu kiến trúc MIPS

Các Thủ Tục Hỗ Trợ Trong Phần Cứng Máy Tính

❖Các khái niệm và tên gọi:

- **Stack pointer** (SP): Địa chỉ của phần tử được cấp phát gần nhất (tức phần tử được cấp phát cuối cùng cho đến thời điểm hiện tại). Trong MIPS, giá trị này lưu trong thanh ghi $sp (29).
- Trong khi đó, thanh ghi $fp ($30) dùng làm con trỏ frame (**Frame pointer**).

Stack gồm nhiều frame; frame cuối cùng trong stack được trỏ tới bởi $sp, frame đầu tiên (cho vùng lưu các thanh ghi cũng như biến cục bộ) được trỏ tới bởi $fp (xem hình slide trước)

- Stack được xây dựng theo kiểu **từ địa chỉ cao giảm dần xuống thấp**, vì thế frame pointer luôn ở trên stack pointer.

Đây là quy ước của MIPS. Tất nhiên stack có thể được xây dựng theo cách khác, tùy vào mỗi môi trường làm việc mà ta tuân thủ các quy ước.

Các Thủ Tục Hỗ Trợ Trong Phần Cứng Máy Tính

❖Các khái niệm và tên gọi:

- **Push**: Việc lưu một phần tử vào stack gọi là Push
- **Pop**: Việc lấy một phần tử ra khỏi stack gọi là Pop

Trong tập lệnh của MIPS chuẩn, không có lệnh Pop và Push. Việc thực hiện này phải do người lập trình thực hiện:

✓Theo quy ước của MIPS, khi PUSH một phần tử vào stack, đầu tiên $sp nên trừ đi 4 byte (1 word) rồi sau đó lưu dữ liệu vào word nhớ có địa chỉ đang chứa trong $sp.

✓Ngược lại, khi POP một phần tử từ stack, dữ liệu được load ra từ word nhớ có địa chỉ đang chứa trong $sp rồi sau đó $sp được cộng thêm 4

Các Thủ Tục Hỗ Trợ Trong Phần Cứng Máy Tính

❖Cấu trúc một thủ tục/hàm con

Ví dụ 2: Một thủ tục chỉ có đủ ba phần: thân hàm, phần đầu và cuối

Viết đoạn code thực hiện

- **Chương trình chính có:**

✓Bốn giá trị a, b, c, d lần lượt chứa trong $s0, $1, $s2, $s3

✓Chương trình truyền bốn tham số này vào thủ tục tên "proc_example" để lấy kết quả của phép toán trên. Sau đó đưa kết quả vào thanh ghi $s6

- **Thủ tục proc_example có:**

✓Bốn input (a, b, c, d)

✓Một output, là kết quả của phép toán: (a + b) – (c + d)

✓Quá trình tính toán sử dụng 2 thanh ghi tạm $t1, $t2 và một thanh ghi $s0

```
        add $a0, $s0, $zero
        add $a1, $s1, $zero
        add $a2, $s2, $zero
        add $a3, $s3, $zero
        jal proc_example

        add $s6, $v0, $zero
        ......
proc_example:
        addi $sp, $sp, -4
        sw $s0, 0($sp)
        add $t0, $a0, $a1
        add $t1, $a2, $a3
        sub $s0, $t0, $t1
        add $v0, $s0, $zero
        lw $s0, 0($sp)
        addi $sp, $sp, 4
        jr $ra
```

Bốn tham số truyền cho thủ tục proc_example trước khi gọi hàm. (Theo quy ước, tham số truyền cho thủ tục phải truyền vào các thanh ghi $a0-$a3)

Sau khi hàm proc_example thực hiện xong, giá trị trả về của hàm theo quy ước chứa trong $v0. Lệnh này chuyển dữ liệu từ $v0 vào thanh ghi $s6

Phần đầu: lưu dữ liệu vào stack cho thủ tục. Do trong phần thân hàm có sử dụng thanh ghi $s0 nên trước khi tính toán, giá trị cũ của $s0 nên được lưu trong stack. Nếu có n thanh ghi cần lưu, mỗi thanh ghi một word, ta cần 4n byte cho stack. Trong trường hợp này, ta chỉ cần 4 byte. Vì stack phát triển theo kiểu từ cao xuống thấp, nên cấp phát 1 frame (1 word cho stack) thì $sp trừ 4. Hai lệnh này còn tương ứng với PUSH một phần tử vào stack

Phần thân: Sau khi tính toán, kết quả trả về phải lưu vào $v0 như quy ước

Phần cuối: Trước khi ra khỏi hàm, giá trị đang lưu của $s0 phải phục hồi lại; và 1 frame của stack phải được xóa đi bằng cách lấy $sp + 4. Hai lệnh này tương ứng với POP một phần tử ra khỏi stack

❖Cấu trúc một thủ tục/hàm con

Trong ví dụ 1, thân thủ tục không sử dụng bất kỳ thanh ghi $s nào, nên không cần khởi tạo stack để lưu các thanh ghi này lại. Trong ví dụ 2, thân thủ tục có sử dụng thanh ghi $s, nên stack phải được khởi tạo để lưu; dẫn đến trước khi kết thúc hàm phải giá trị đã lưu phải trả về lại các thanh ghi và xóa stack.

(Lưu ý: Code trong ví dụ 2 có thể viết lại mà không cần dùng thanh ghi $s0; nhưng ví dụ này cố tình dùng $s0 để thấy được việc lưu trên stack phải thực hiện như thế nào)

Các Thủ Tục Hỗ Trợ Trong Phần Cứng Máy Tính

Tóm lại:

- Với một thủ tục/hàm con:
  - ✓ Input: $a0, $a1, $a2, $a3
  - ✓ Output: $v0, $v1
  - ✓ Địa chỉ quay về được lưu trong $ra
- Stack được sử dụng trong các trường hợp sau
  - ✓ Nếu thủ tục (hàm con) cần nhiều hơn bốn thanh ghi để làm tham số hoặc nhiều hơn hai thanh ghi làm giá trị trả về thì không gian stack sẽ được dùng hỗ trợ trong trường hợp này.
  - ✓ Nếu thủ tục có sử dụng một số thanh ghi loại $s, thì theo quy ước của MIPS, trước khi thực hiện thủ tục, giá trị của các thanh ghi này phải được lưu trữ vào stack và được hồi phục lại khi kết thúc thủ tục *(Các thanh ghi $t không cần lưu)*.

Các Thủ Tục Hỗ Trợ Trong Phần Cứng Máy Tính

Nested procedure:

❖ Các thủ tục có thể gọi lồng vào nhau.

❖ Các thủ tục mà không gọi các thủ tục khác là các thủ tục lá (leaf procedures). Ngược lại là nested procedures.

❖ Chúng ta cần cẩn thận khi sử dụng các thanh ghi trong các thủ tục, càng cần phải cẩn thận hơn khi gọi một nested procedure.

Tổng kết:

- Tuần 3, 4, 5 và 6 đã trình bày các nội dung liên quan đến hợp ngữ của MIPS, biết như thế nào để lập trình một chương trình dùng assembly.

- Với một chương trình assembly dài, có nhiều chức năng, như lập trình ngôn ngữ cấp cao, mỗi chức năng có thể đưa vào một thủ tục (procedure) (như các hàm hoặc hàm con trong ngôn ngữ cấp cao)

- Khi lập trình với procedure cho MIPS, các điểm cần chú ý: stack, $sp, $fp và cách truyền tham số.

Tuần 5 – Kiến trúc bộ lệnh

❖ **Lý thuyết: Đọc sách tham khảo**

- Mục: 2.8
- Sách: *Computer Organization and Design: The Hardware/Software Interface*, Patterson, D. A., and J. L. Hennessy, Morgan Kaufman, Revised Fourth Edition, 2011.

❖ **Bài tập:** file đính kèm