Lecture-7

Function

Unconditional Jump:

(1) jr $ra [for return]

Jump and Link → (II) jal L [for call]

(III) j L

(*) Function এর কাজ করতে কিছু register দরকার হবে।:-

এযন পর্যন্ত ৩ ধরনের register দেখেছি:

(1) Save register

(2) Temporary register

(3) Zero register

(*) আমাকে আরো আরো কিছু register এর সাথে পরিচিত হবো।-

→এর মধ্যে একটি হলো Argument register.

[Note: Function call করলে যে value গুলো pass করি গুলো হচ্ছে parameter value.]

X = Func(a, b);
parameters

* parameters গুলোকে যে variables গুলো Recieve করে তাদেরকে বলা হয় Argument value.

int func(int x, int y){ }
Arguments

* Arguments variable গুলোকে কখনোই saved register এ রাখা হবে না।

→ Argument Register: $a0, $a1, $a2, $a3

→ Return Value Register: $v0, $v1
[Function এর value Return এর জন্য use করা হয়।]

→ Return Address: $ra
[

→ Stack pointer: $sp

(*) jr $ra: আমরা Function call করার পর। Function value return করে। তখন code টি jump করে যেখান থেকে call হয়েছিল সেখানে ফিরে যায়। এই unconditional jump করে যেখান থেকে call হয়েছে সেখানে return হওয়ার কাজটা আমরা: `jr $ra` এর দ্বারা করি।

→ এখানে এই register এর কাজ হচ্ছে jump করে return করা এই $ra register এর value যেখানে সেখানে।

→ এখন return Address ($ra) register এর value কত হবে। সেটা Main Function করে। কিভাবে করে। এখন সেটা আলোচনা করব।

jal: (jump and link):

এর কাজ হচ্ছে Function call হওয়ার পরে যে Address এ যাবে। সেই Address এর সাথে connect করা।

ex: jal 4000 // এর মানে হচ্ছে call থেকে jump করে 4000 এ jump করে এবং link হবে। এবং link টি main

method এর সাথে
করে।

So Jal 2টি কাজ করে:-

(I) Jump to called Address

(II) Link to main method:

Link এর কাজ: Main Function এর $pa এর সাথে Link করে

$pa = (P.C.) + 4

Pragram Counter: Main Function এর প্রথম line থেকে count করা শুরু হয়। এটা প্রতি line এর count রাখে।

→ $pa এর ক্ষেত্রে P.C স্বাভাবিকভাবে যে line এ call হয়েছে সে line এর Address এর count save রাখে এটা পরবর্তীতে $pa এর সাথে যোগ হয়। example:

P.C → 2012: c = add(a+b);

So, Here,

P.C = 2012

$\therefore \$ra = pc + 4 = 2012 + 4 = 2016$

$\therefore$ jr $ra // এর মানে 2016 No. Address jump করে ফিরে যাবে।

এবং, যদি 4000 // এবং এই যদি মূলত 4000 এ jump করে এবং $ra(2016) এর মধ্যেই link তৈরি করে।

Summary

(*) jr: (jump register)

- Unconditional jump to the address specified in a register.

(*) jal: (jump and link)

- Jump: jump to the function_Address
- Link: saves P.C. (Program counter) + 4 in register $ra

✳ The calling program, or caller, puts the parameter values in ($a0 – $a3) and uses jal X to jump to procedure X (sometime named callee).

The callee then performs the calculations, places the result in $v0 – $v1, and returns control to the caller using [jr $ra].

Now solve a code and understand it in details.

✳ Example:

```
int function(int g, int h, int i, int j)
{ int f, a, b=20, c=30;
  a=b+c;
  f=(g+h)-(i+j);
  f=a+f;
  return f; }
```

f = $s0 | a = $s1 | b = $s2 | c = $s3

Solution:

- Solution (Partial):

```
addi $s2, $zero, 20
addi $s3, $zero, 30
add $s1, $s2, $s3
add $t0, $a0, $a1
add $t1, $a2, $a3
sub $s0, $t0, $t1
add $s0, $s1, $s0
add $v0, $s0, $zero
```

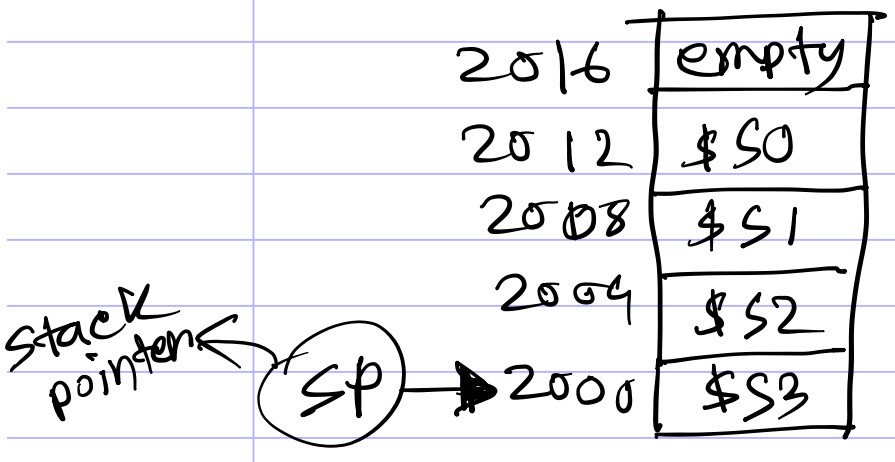
[partial বলতে এখানে used local variable গুলো main function এও use হতে পারে। যার জন্য main function এর value গুলো এখানে overwrite হয়ে যাওয়ার সম্ভাবনা আছে।

যার জন্য partial solution এ আমরা আগে Solution (Before) করব যেখানে partial এর local variable এর register গুলো stack এ push করে আসব।]

Now lets understand How to do solution Before and push in stack part:

① Stack push এর সময় যত গুলো variable (n গুলো) আছে। Stack টাকে কি (4*n) গুলো ঘর নিচে নামাতে।

② Stack এর প্রথম ঘর টা empty থাকে।



③ এখন Stack এ push করব অর্থাৎ stack এ write করব। so left to right variable গুলোকে

int a, b, c;

Solution (Before):

- addi $sp, $sp, -16 // ২টি variable এর 16 byte stack pointer নামাবো।

16 no ঘর empty

- sw $s0, 12($sp)
- sw $s1, 8($sp)
- sw $s2, 4($sp)
- sw $s3, 0($sp)

→ Spilling register

(*) Now function এর কাজ শেষ হয়ে যাওয়ার পর stack টা ফ্রি করতে হবে। তার জন্য solution(After) use করব।

→ solution After এ pop() এর কাজ করব।

এই pop() এর কমান্ড নিচে order এ করব।

Solution (After):

- lw $s0, 0($sp)
- lw $s1, 4($sp)
- lw $s2, 8($sp)
- lw $s3, 12($sp)
- addi sp, sp, +16 // একত্রে 16 ঘর জায়গা ফেরত করে দিব।
- jr $ra // Function এর কাম শেষ এবার যেখান থেকে call হয়েছে সেখানের অ্যাড্রেস ধরে return করবে।

* Now Overall Solution *

↓

Solution (Before):

- addi $sp, $sp, -16

16 no sa empty

- sw $s0, 12($sp)
- sw $s1, 8($sp)
- sw $s2, 4($sp)
- sw $s3, 0($sp)

- Solution (Partial):

```
addi $s2, $zero, 20
addi $s3, $zero, 30
add $s1, $s2, $s3
add $t0, $a0, $a1
add $t1, $a2, $a3
sub $s0, $t0, $t1
add $s0, $s1, $s0
add $v0, $s0, $zero
```

Solution (After):

- lw $s3, 0($sp)
- lw $s2, 4($sp)
- lw $s0, 8($sp)
- lw $s1, 12($sp)
- addi sp, sp, +16
- jr $ra

Ans

✳ Example:

```
int add(int c, int d)
{
    int f, a=20, b=10;
    f = a+b+c+d;
    return f;
}
```

f = $s0 | a = $s1 | b = $s2 | c = $s3

Solution:

Solution (before):

```
addi  $sp, $sp, -12
sw    $s0, 8($sp)
sw    $s1, 4($sp)
sw    $s2, 0($sp)
```

Solution (partial):

```
addi  $s1, $s1, 20
addi  $s2, $s2, 10

add   $t0, $s1, $s2
add   $t1, $a0, $a1
add   $s0, $t0, $t1

add   $v0, $s0 $zero // v0 = f+0
```

Solution (After):

```
lw   $s0, 0($sp)
lw   $s1, 4($sp)
lw   $s2, 8($sp)
addi $sp, $sp, +12
jr   $ra
```