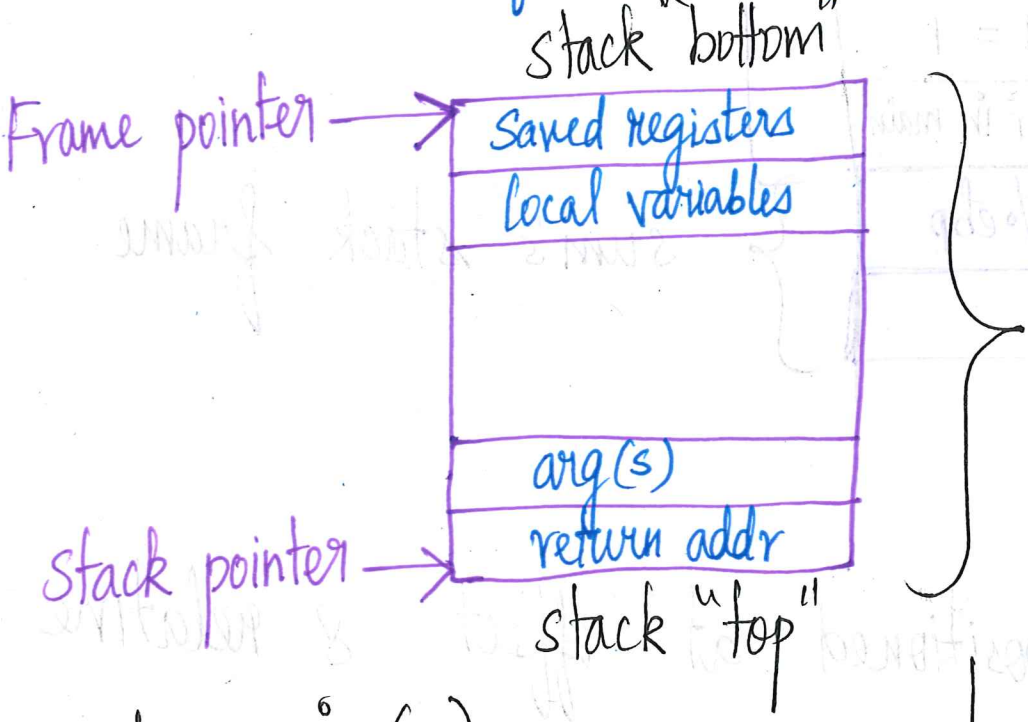
Feb 29, 2016 Lecture - 17

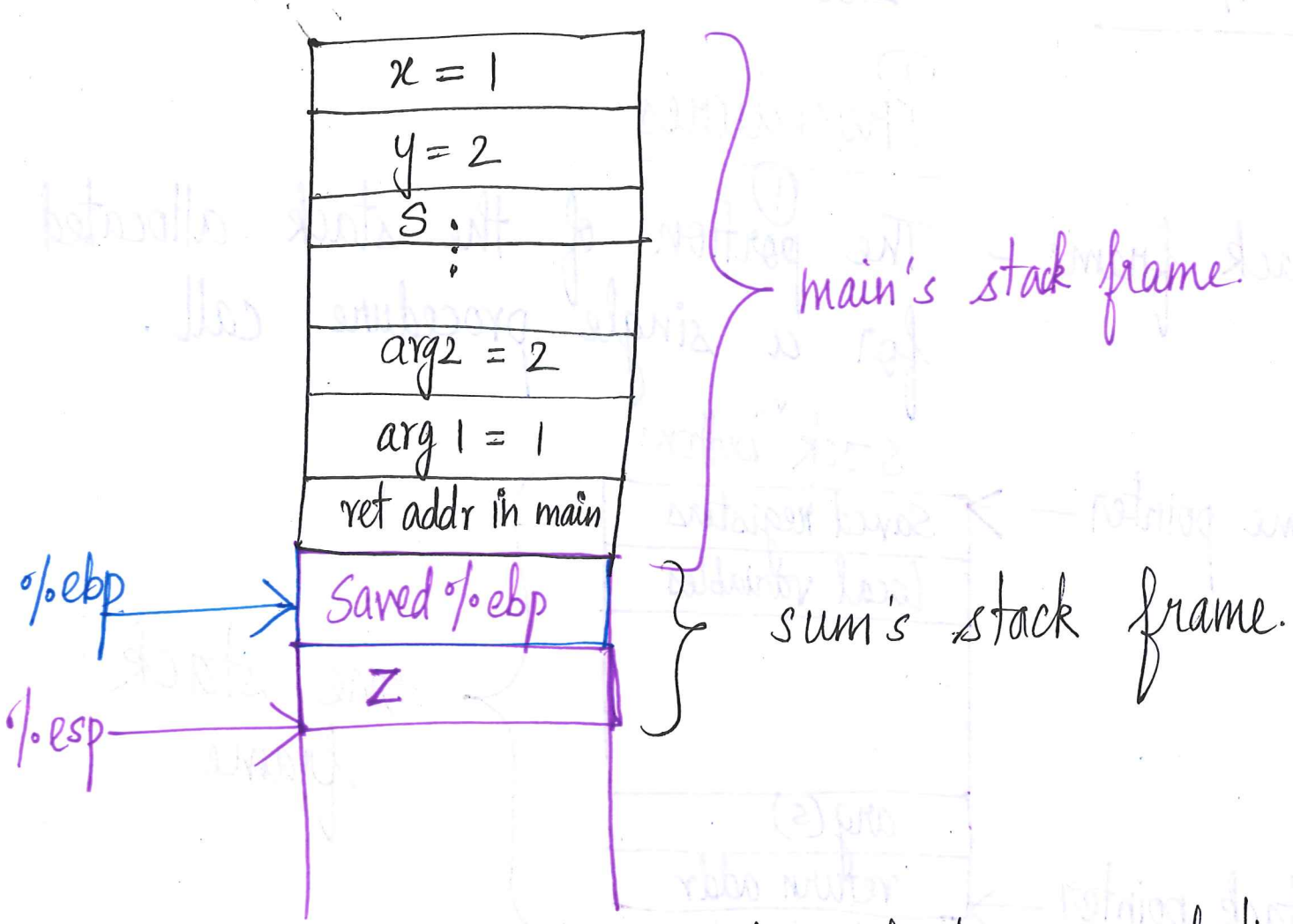
Procedures

Stack frame - The portion of the stack allocated for a single procedure call.




```
int main()
{
    int x = 1;
    int y = 2;
    int s = sum(1, 2);
    printf("s = %d", s);
    return 0;
}
```

```
int sum(int x,
        int y)
{
    int z;
    z = x + y;
    return z;
}
```



1st arg is positioned at offset 8 relative to %ebp.

Arg i is at offset $4 + 4i$ relative to %ebp.

$i = 1, 2, \ldots$

For args $\geq$ 4 bytes, larger regions on stack is used.

Transferring Control

Instruction	Desc.
call Label	Procedure Call
call *Operand	"
leave	Prepare stack for return.
ret	Return from call.

call

1. push a return address on the stack.
2. jump to the start of the called ~~prodec~~ procedure.

return address - address of the instruction immediately following the call.

ret

1. pops an address off the stack.
2. jumps to that address.

leave - prepare the stack for returning.

```
movl %ebp, %esp
```
set stack pointer to beginning of frame.

```
popl %ebp
```
Restore saved %ebp and set stack ptr to end of caller's frame.

Register Usage Conventions

Caller - save registers - %eax, %edx, %ecx.
main() (caller) must save these registers.

main() - caller
↓
sum() - callee.

Callee - save registers - %ebx, %esi, %edi
sum() (callee) must save these registers.

eg.
```
int P(int x)
{   int y = x * x;
    int z = Q(y);
    return y + z;
}
```

1. y can be saved in P's stack frame.
2. y can be saved in a callee-save register, in which case Q must make sure that it should save & restore the value of y.