Recitation4

Today @ 12pm

NVIDIA Announcement

Outline

Announcements

Stack!

Optimizations 101

Announcements

Datalab at ECE Course Hub

Buflab due this Thursday

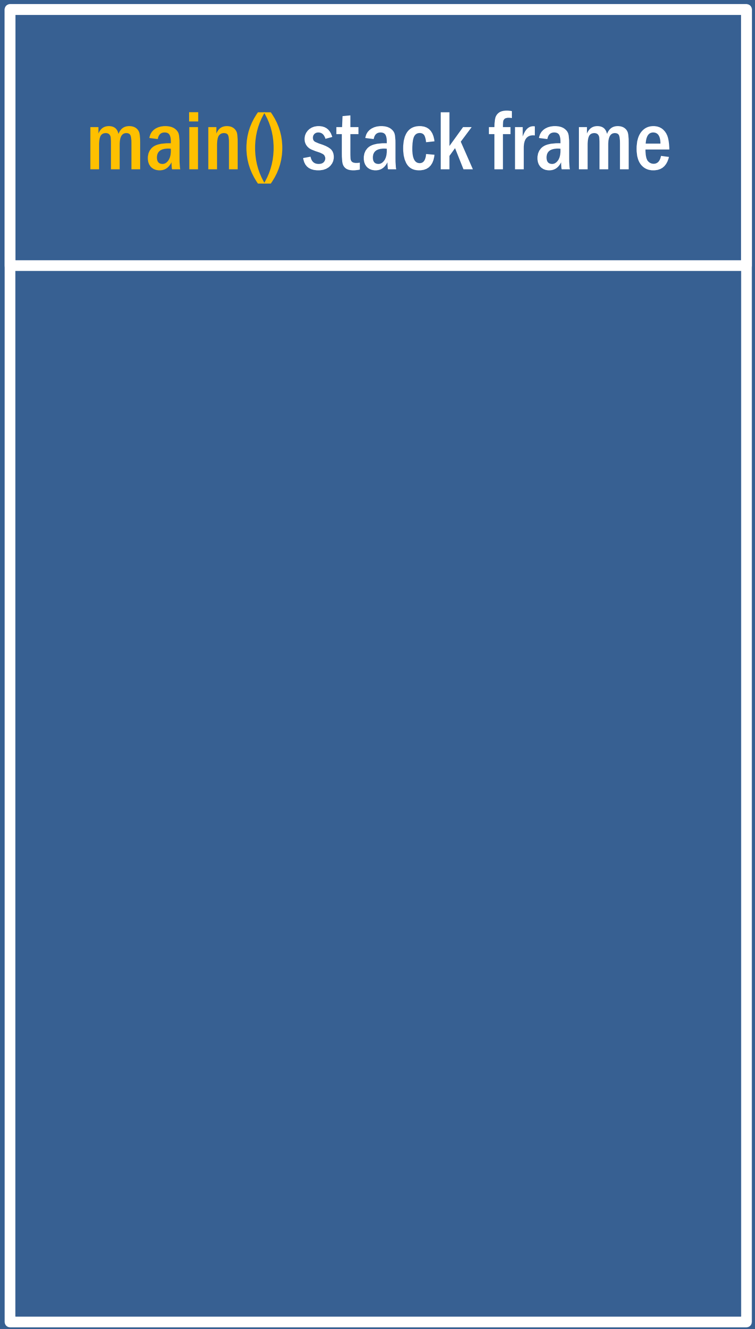
Exam1 in class next Tues.

Stack!

Why is it called ‘stack’?

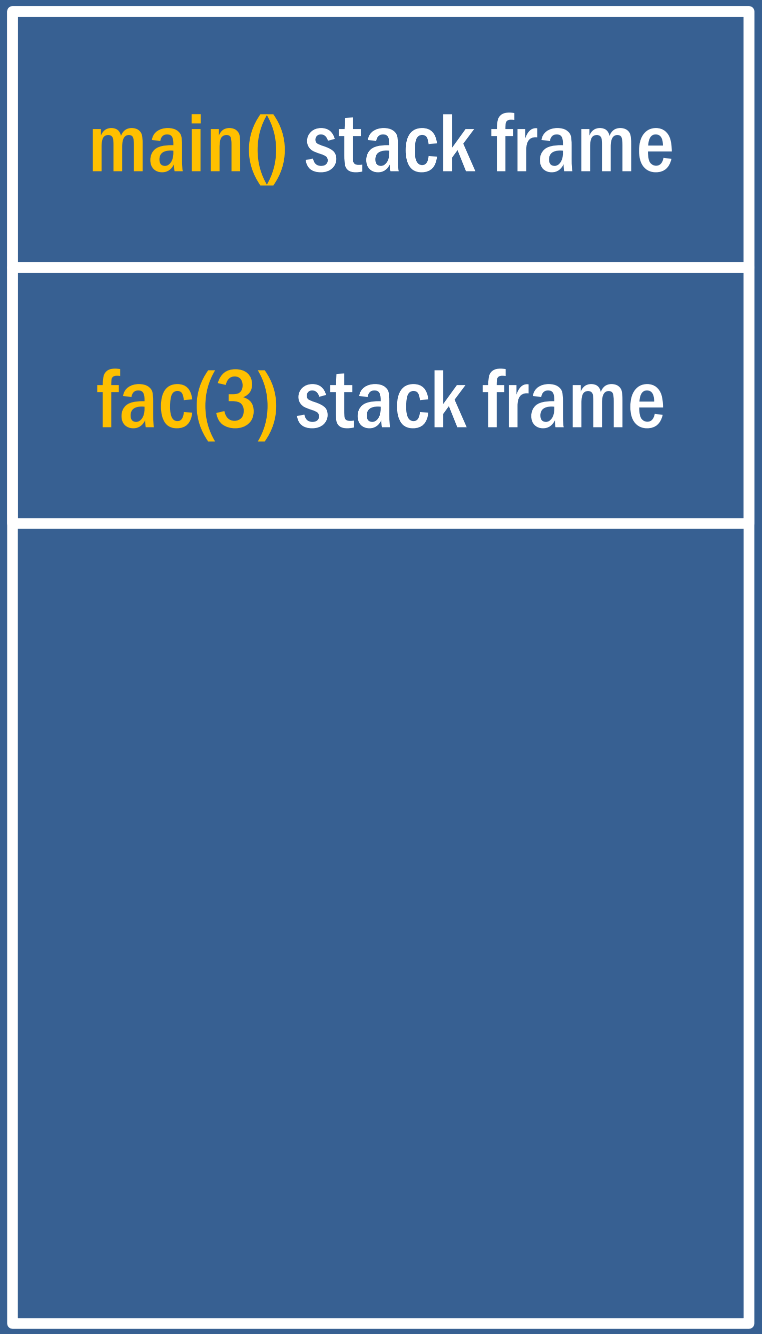
```
int fac(int x)
{
  if (x == 1) return 1;

  return x * fac(x-1);
}


int main()
{
   return fac(3);
}
```

```
int fac(int x)
{
  if (x == 1) return 1;

  return x * fac(x-1);
}


int main()
{
   return fac(3);
}
```



main() stack frame

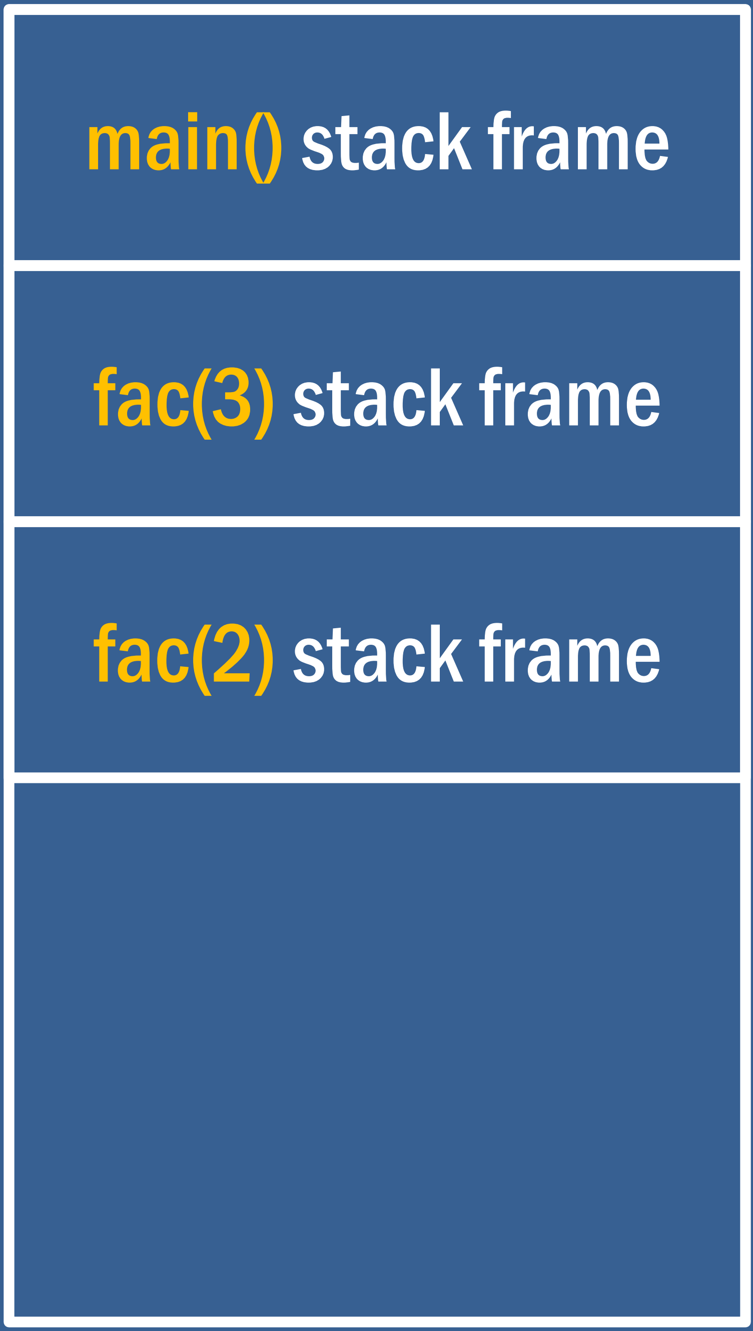
```c
int fac(int x)
{
  if (x == 1) return 1;

  return x * fac(x-1);
}


int main()
{
   return fac(3);
}
```



main() stack frame

fac(3) stack frame

```
int fac(int x)
{
  if (x == 1) return 1;

  return x * fac(x-1);
}


int main()
{
   return fac(3);
}
```

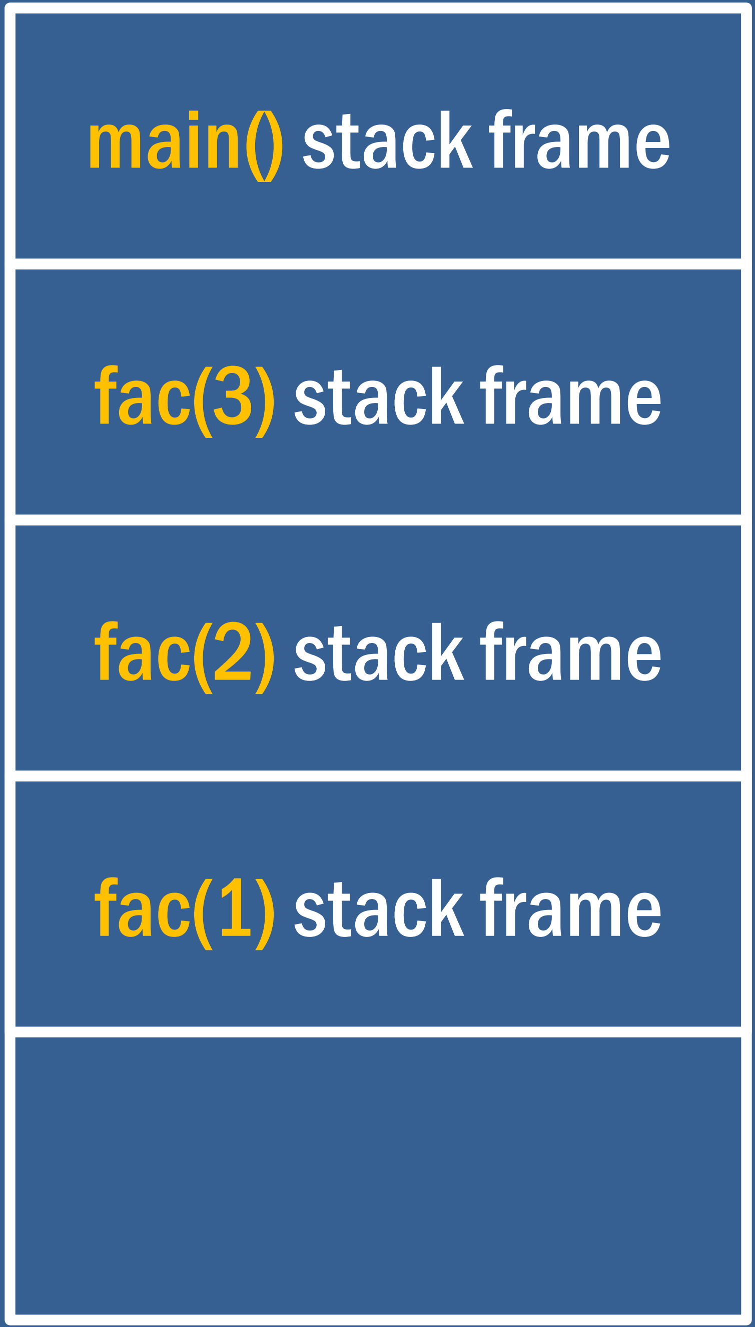


main() stack frame

fac(3) stack frame

fac(2) stack frame

```
int fac(int x)
{
  if (x == 1) return 1;

  return x * fac(x-1);
}


int main()
{
   return fac(3);
}
```



main() stack frame

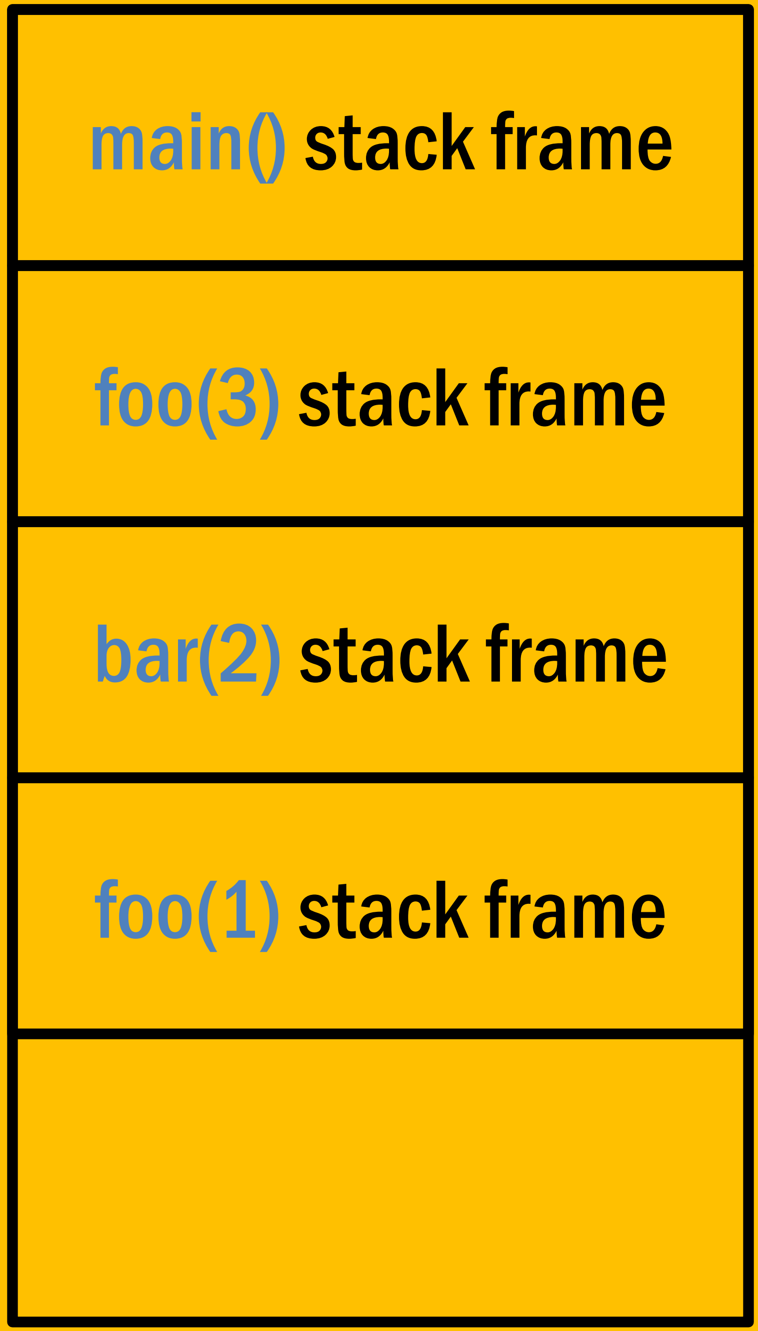
fac(3) stack frame

fac(2) stack frame

fac(1) stack frame

```c
int bar(int x)
{
  if (x == 1) return 1;
  return x * foo(x – 1);
}

int foo(int x)
{
  if (x == 1) return 1;
  return x * bar(x-1);
}

int main()
{
   return foo(3);
}
```

```
int bar(int x)
{
  if (x == 1) return 1;
  return x * foo(x - 1);
}

int foo(int x)
{
  if (x == 1) return 1;
  return x * bar(x-1);
}

int main()
{
   return foo(3);
}
```



main() stack frame

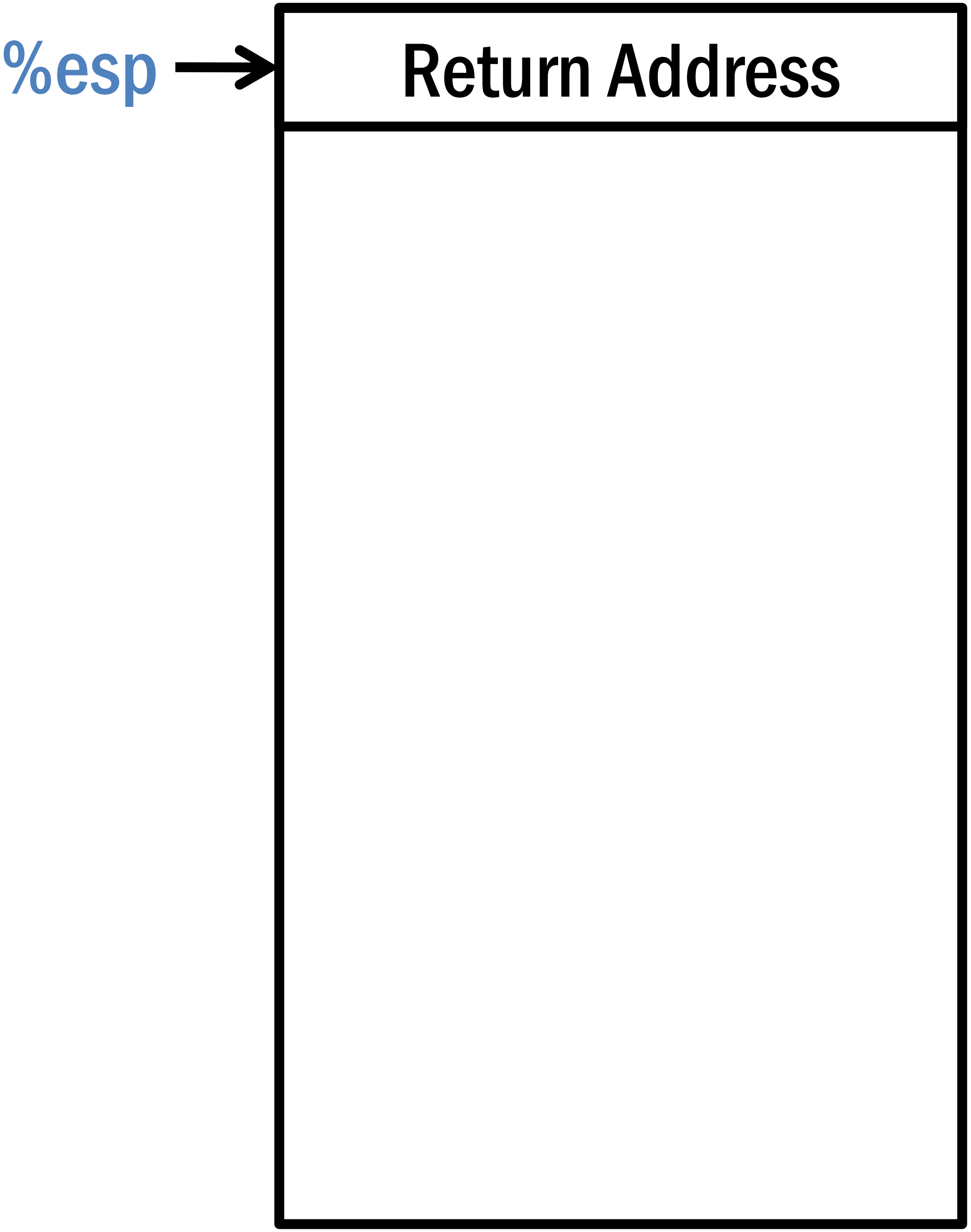
foo(3) stack frame

bar(2) stack frame

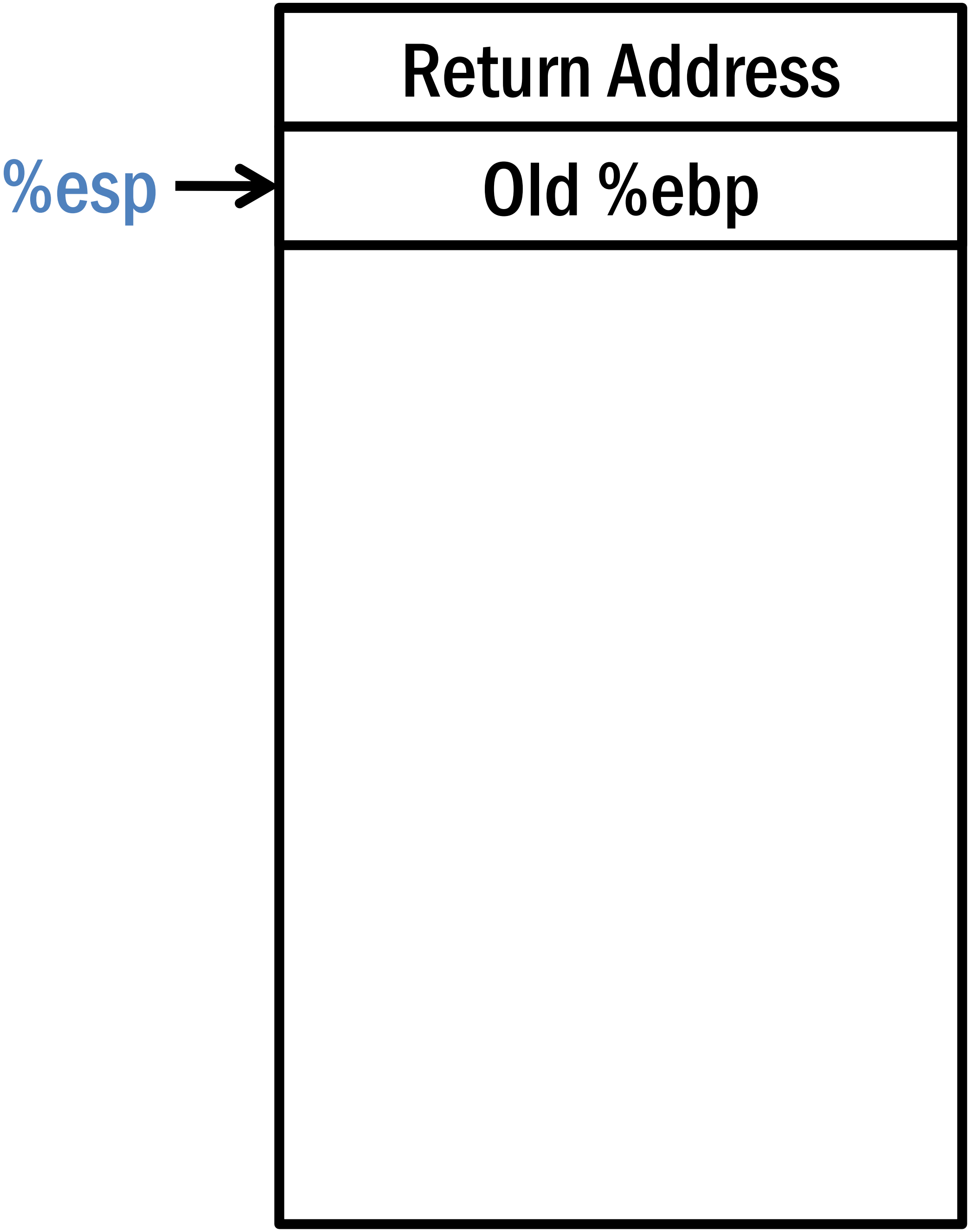
foo(1) stack frame

```
int getbuf()
{
  char buf[32];
  Gets(buf);

  return 1;
}
```

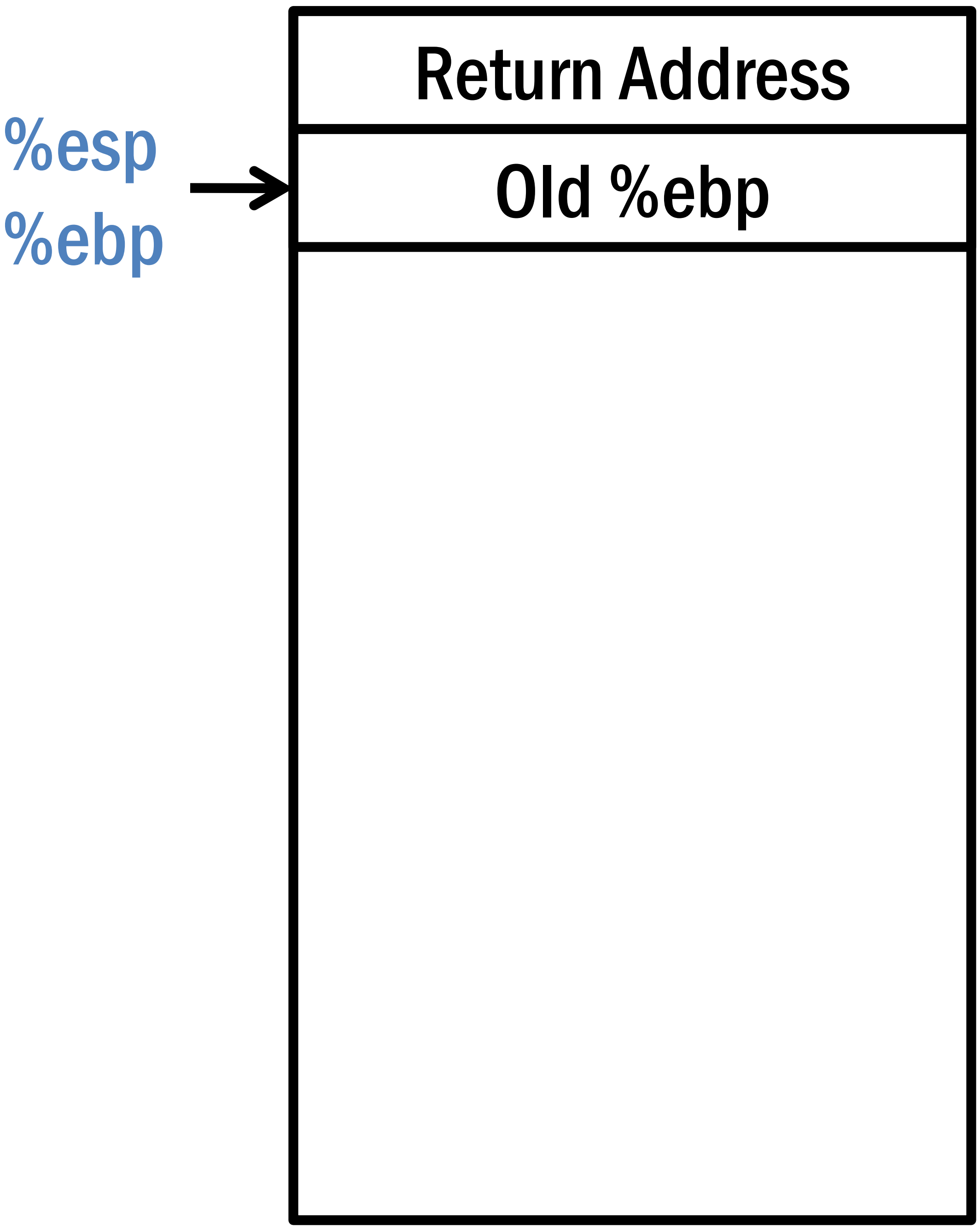
```
int getbuf()
{
   char buf[32];
   Gets(buf);

   return 1;
}
```



```
int getbuf()
{
   char buf[32];
   Gets(buf);

   return 1;
}
```



```
int getbuf()
{
   char buf[32];
   Gets(buf);

   return 1;
}
```



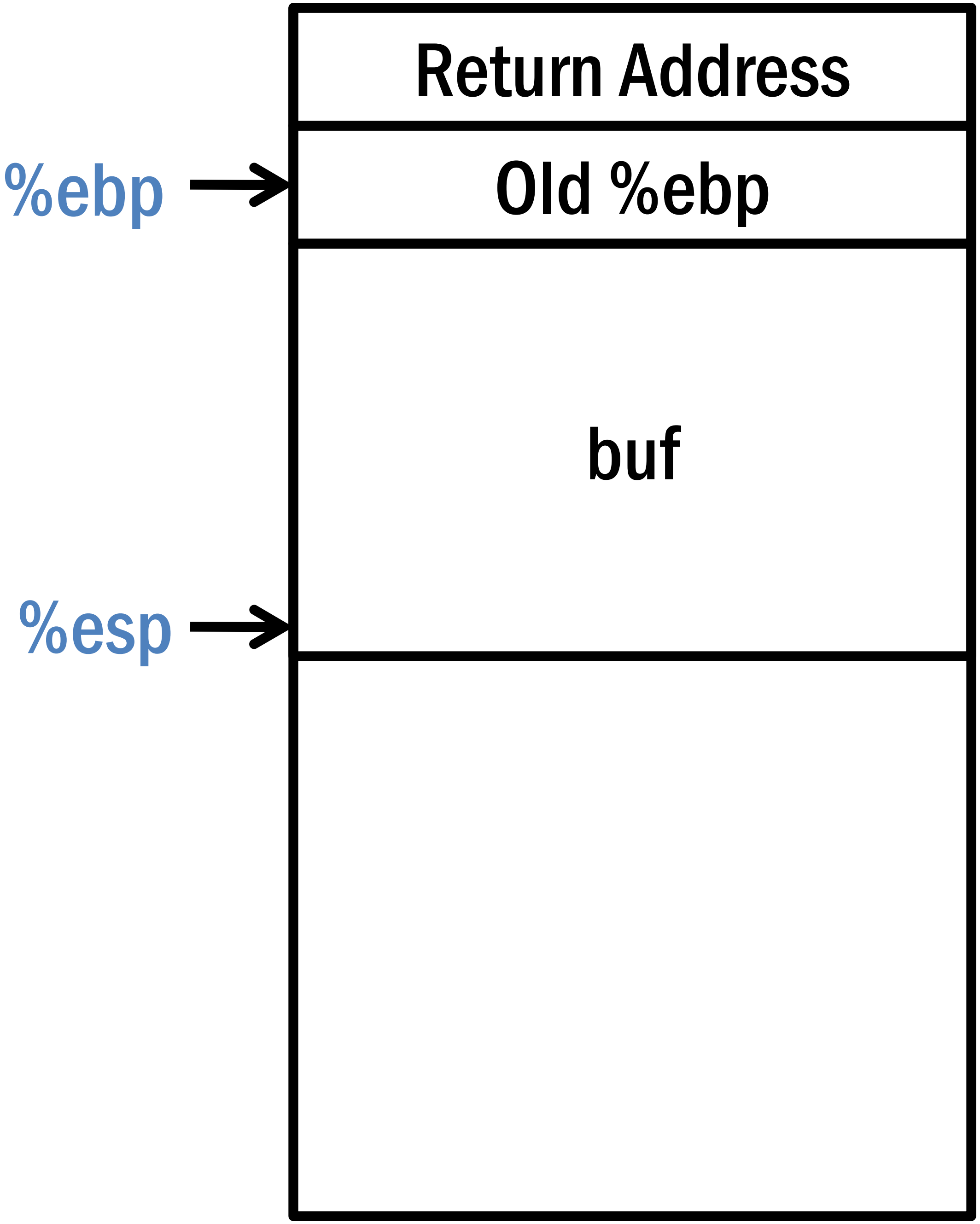
%esp

%ebp

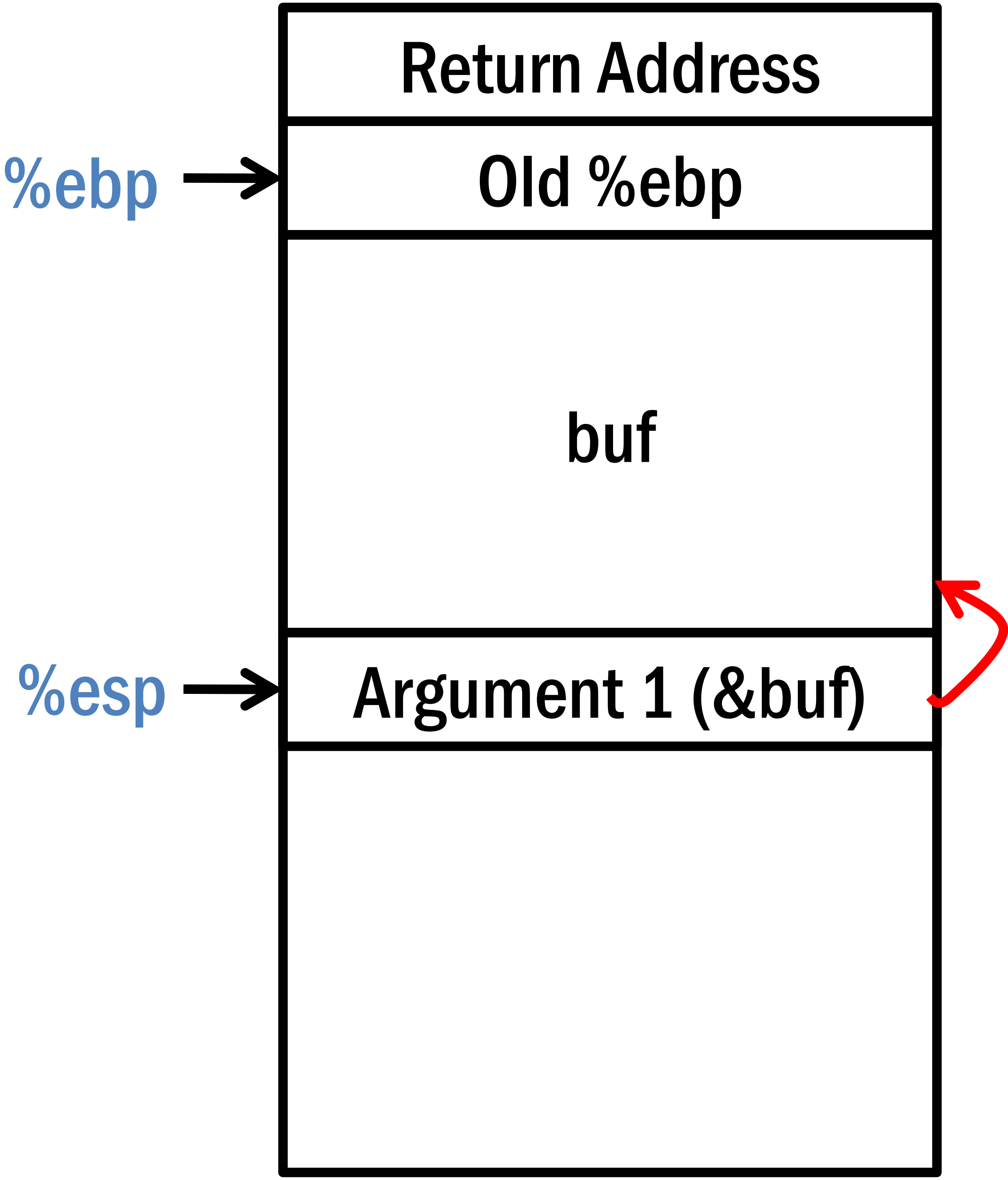
Return Address

Old %ebp

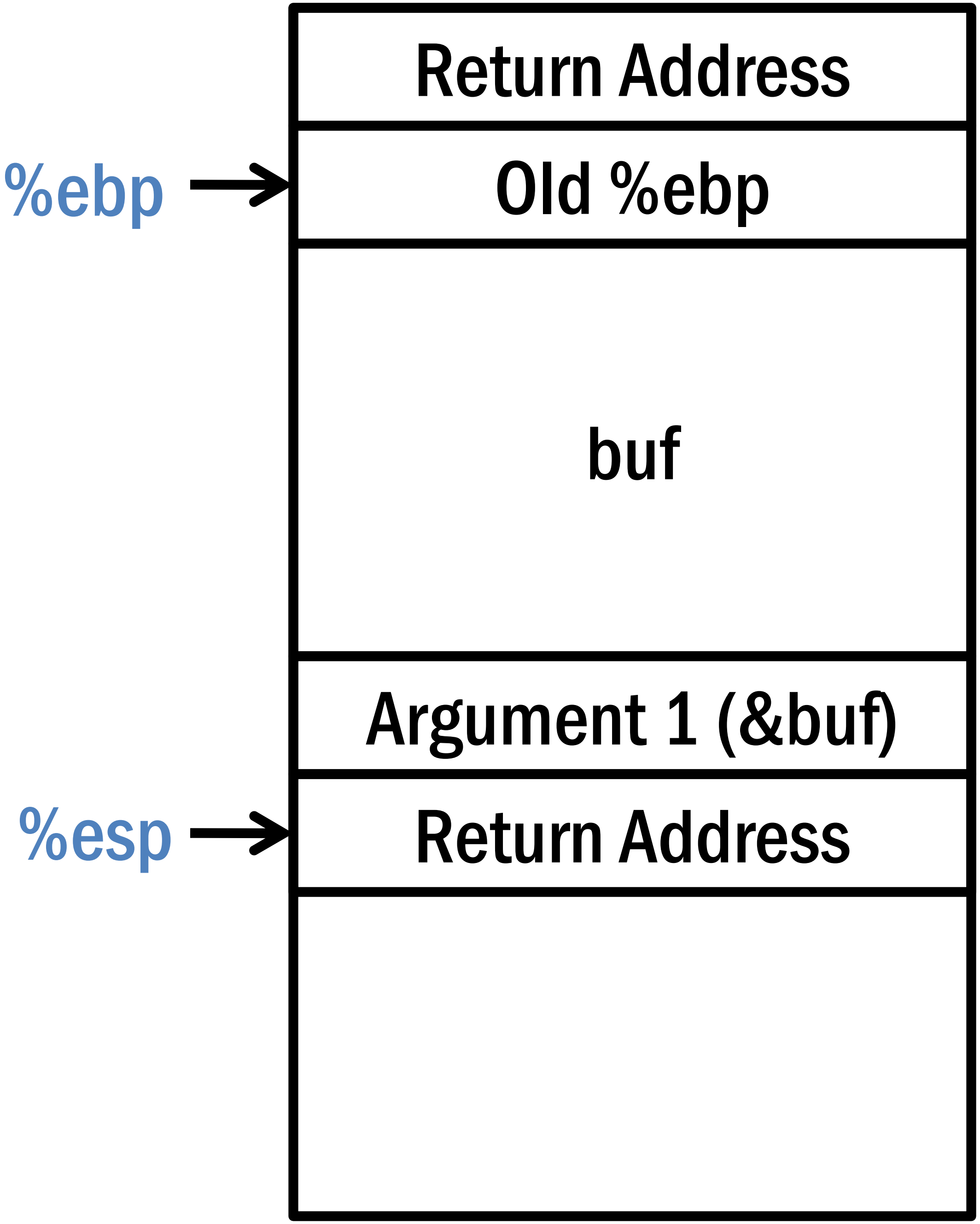
```
int getbuf()
{
   char buf[32];
   Gets(buf);

   return 1;
}
```



```
int getbuf()
{
   char buf[32];
   Gets(buf);

   return 1;
}
```



```
int getbuf()
{
   char buf[32];
   Gets(buf);

   return 1;
}
```



Return Address

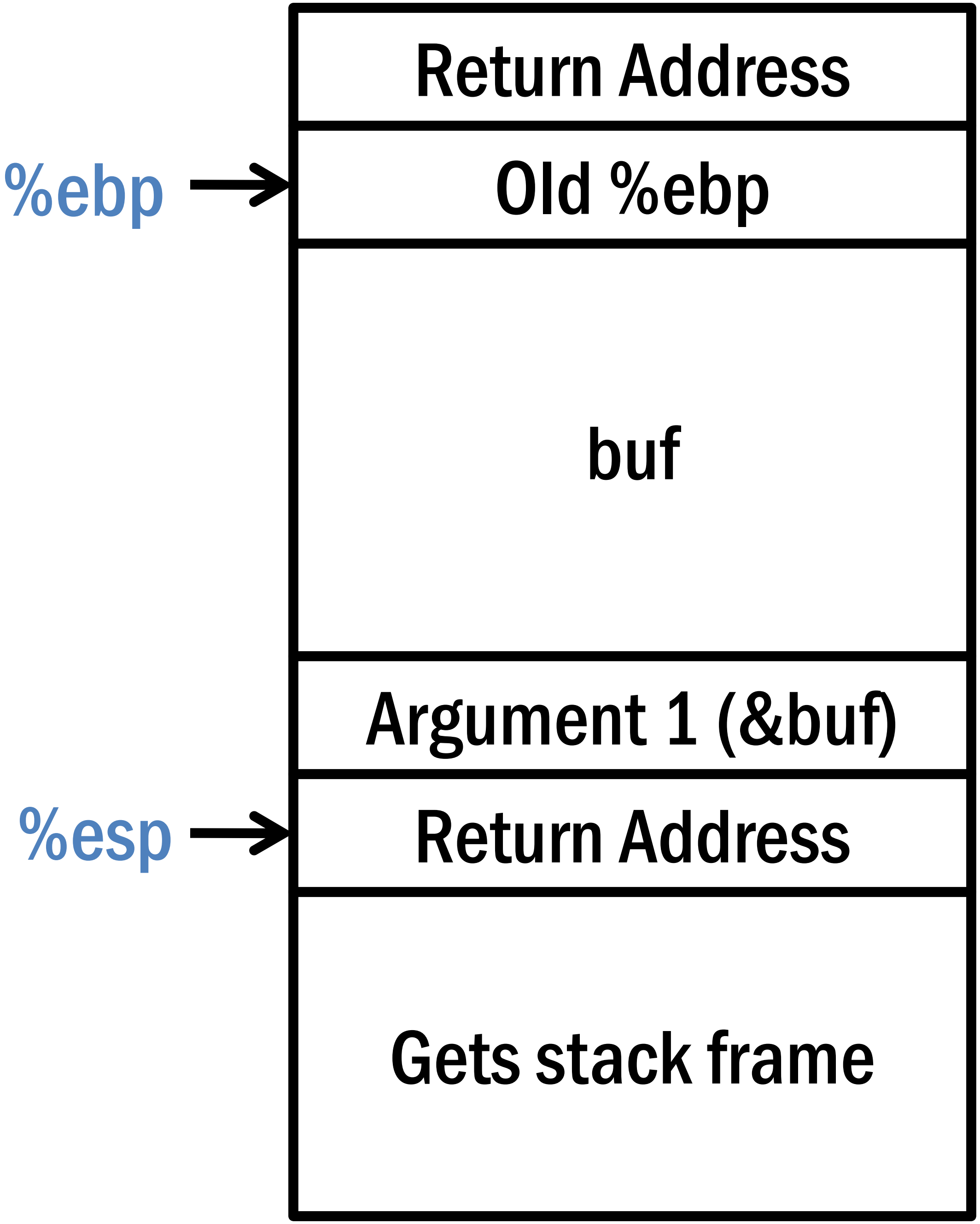
%ebp → Old %ebp

buf

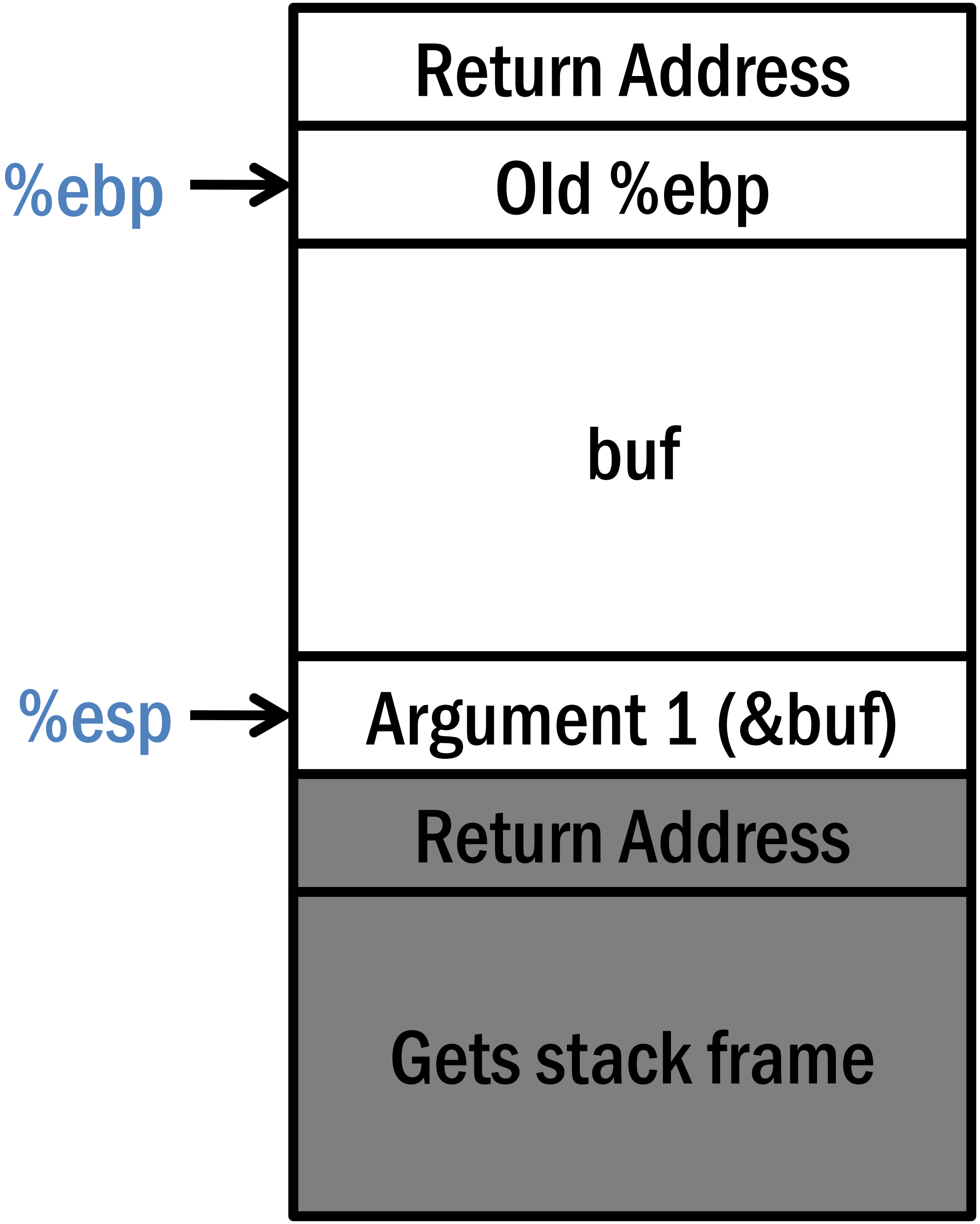
Argument 1 (&buf)

%esp → Return Address

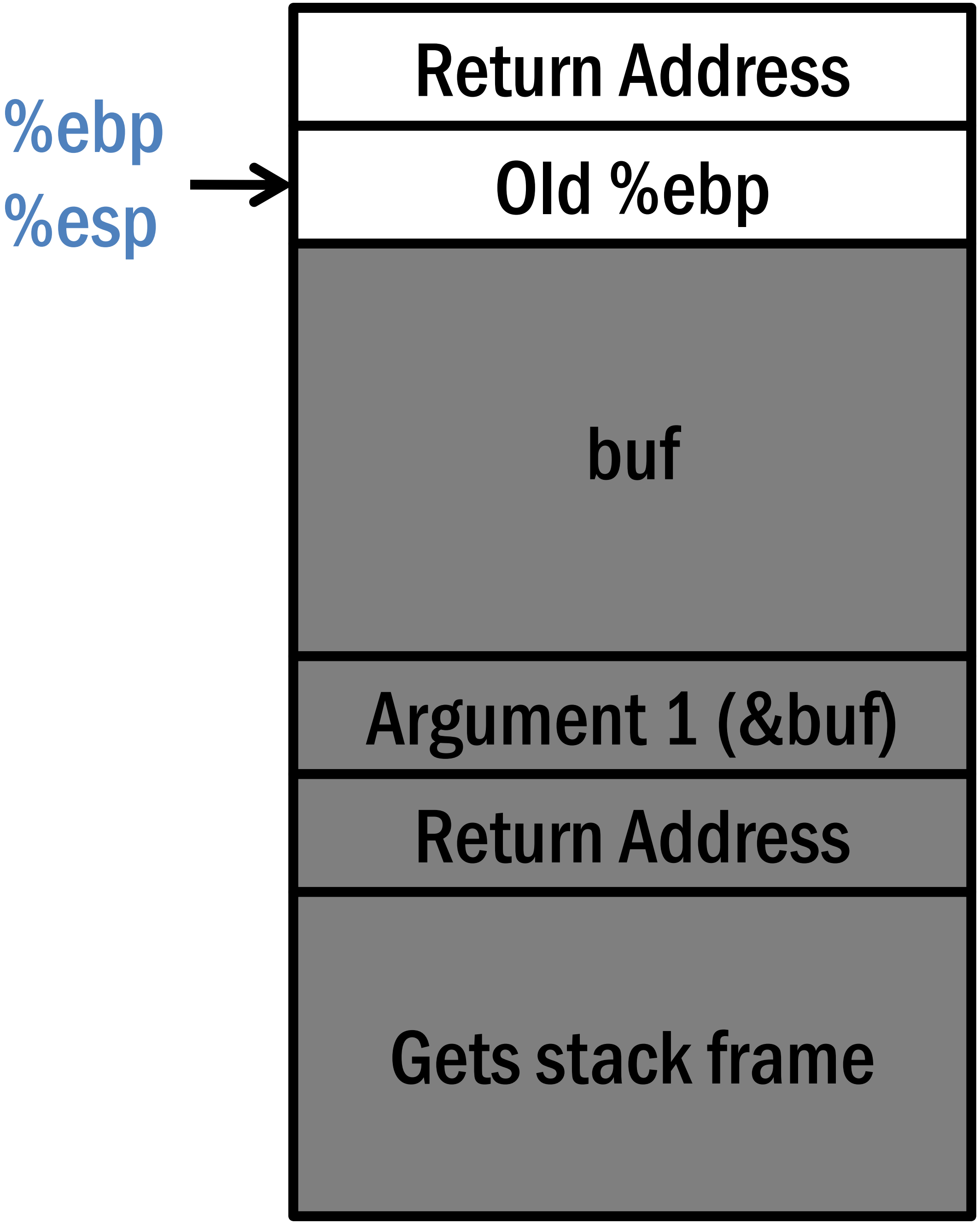
```
int getbuf()
{
   char buf[32];
   Gets(buf);

   return 1;
}
```



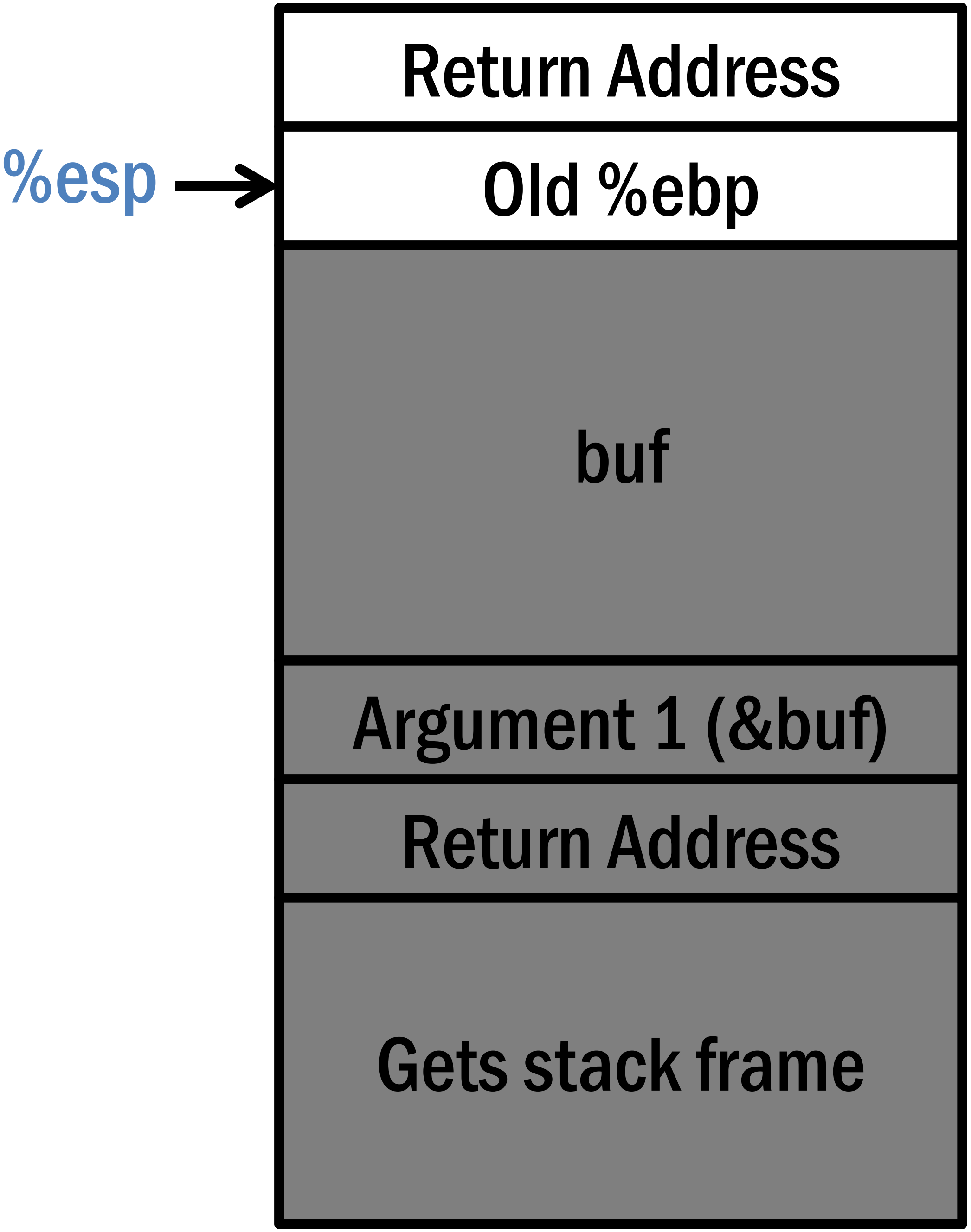
```
int getbuf()
{
   char buf[32];
   Gets(buf);

   return 1;
}
```



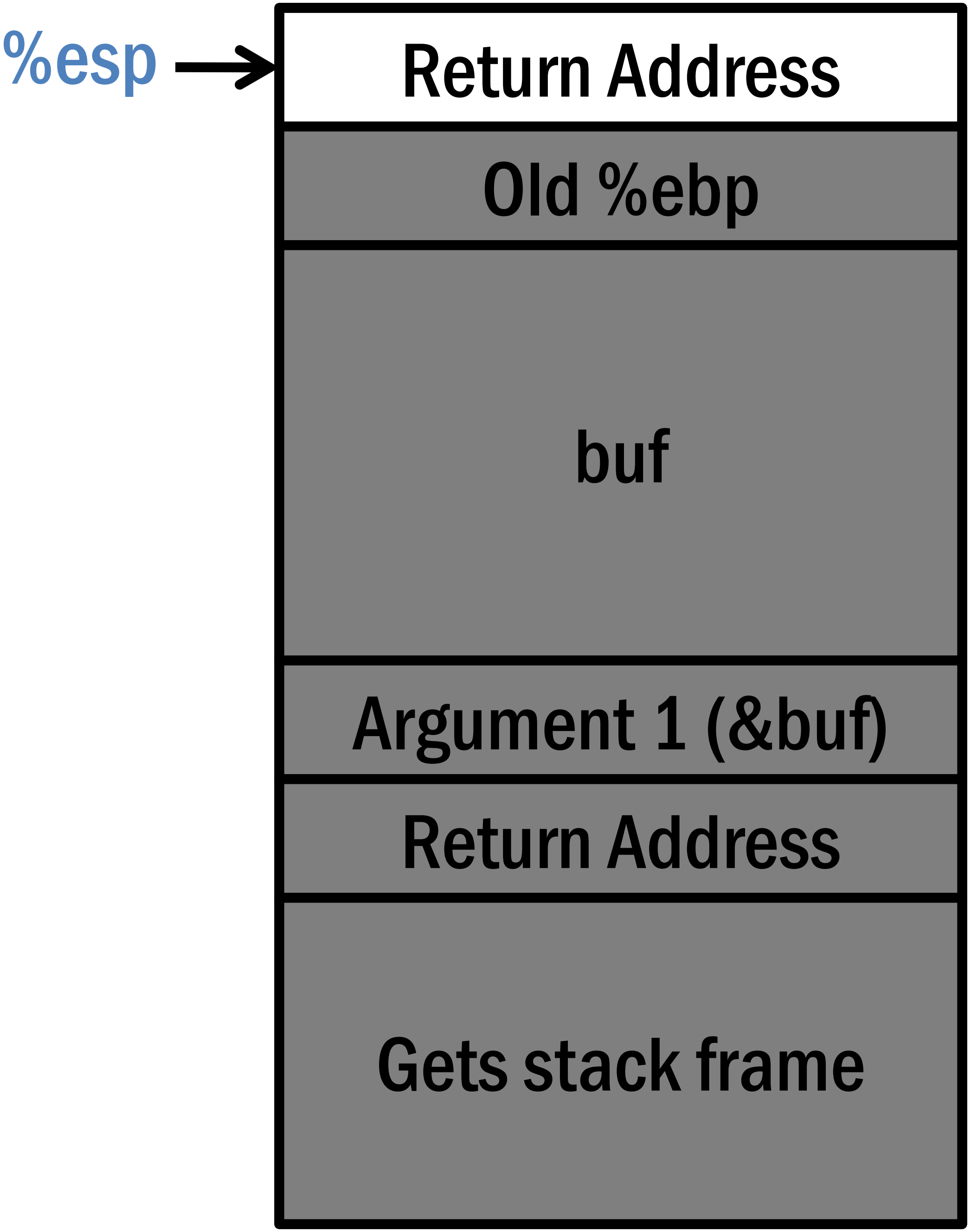
```
int getbuf()
{
   char buf[32];
   Gets(buf);

   return 1;
}
```



```
int getbuf()
{
   char buf[32];
   Gets(buf);

   return 1;
}
```



```
int getbuf()
{
   char buf[32];
   Gets(buf);

   return 1;
}
```



What does the stack allow a programmer to do?

What is the difference between return value vs. return address?

How are function arguments implemented in x86 vs. x86_64?

How can we go about not using a base pointer (%rbp) in x86_64?

Which levels of a computer system are aware of the stack?

(Processor, Compiler, Programming Language)

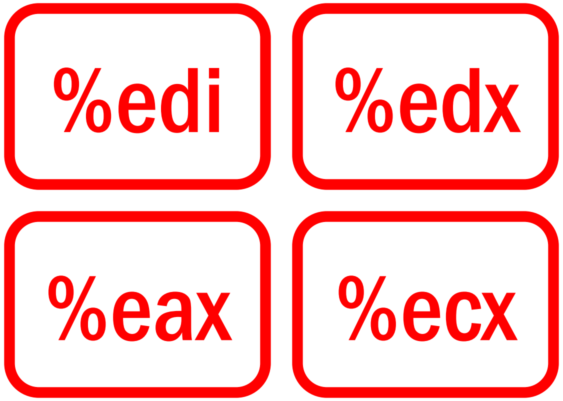
Apple Interview Question

Write a function which can tell whether the stack grows up or down

```
int* helper()
{
  int c = 1;
  return &c;
}

int main()
{
  int a = 1;
  int *b = helper();

  if (&a > b) printf (“Down\n”)
  else printf(“Up\n”);

  return 0;
}
```

Optimizations 101

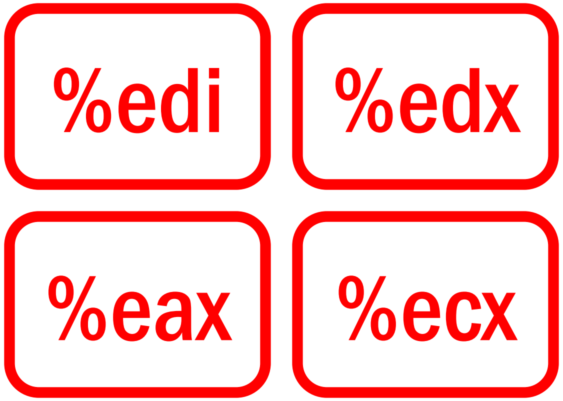
Where is data stored in a program?



%edi %edx

%eax %ecx

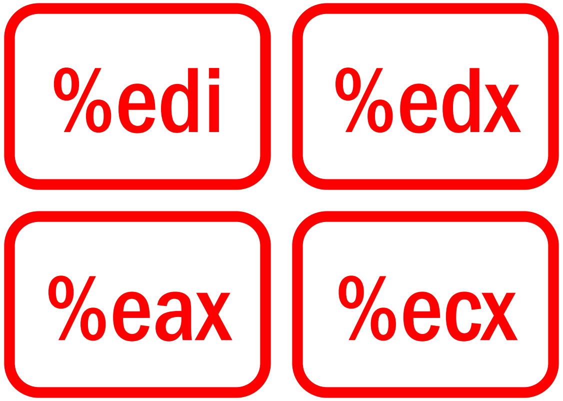
Registers



%edi
%edx
%eax
%ecx

Registers

Memory

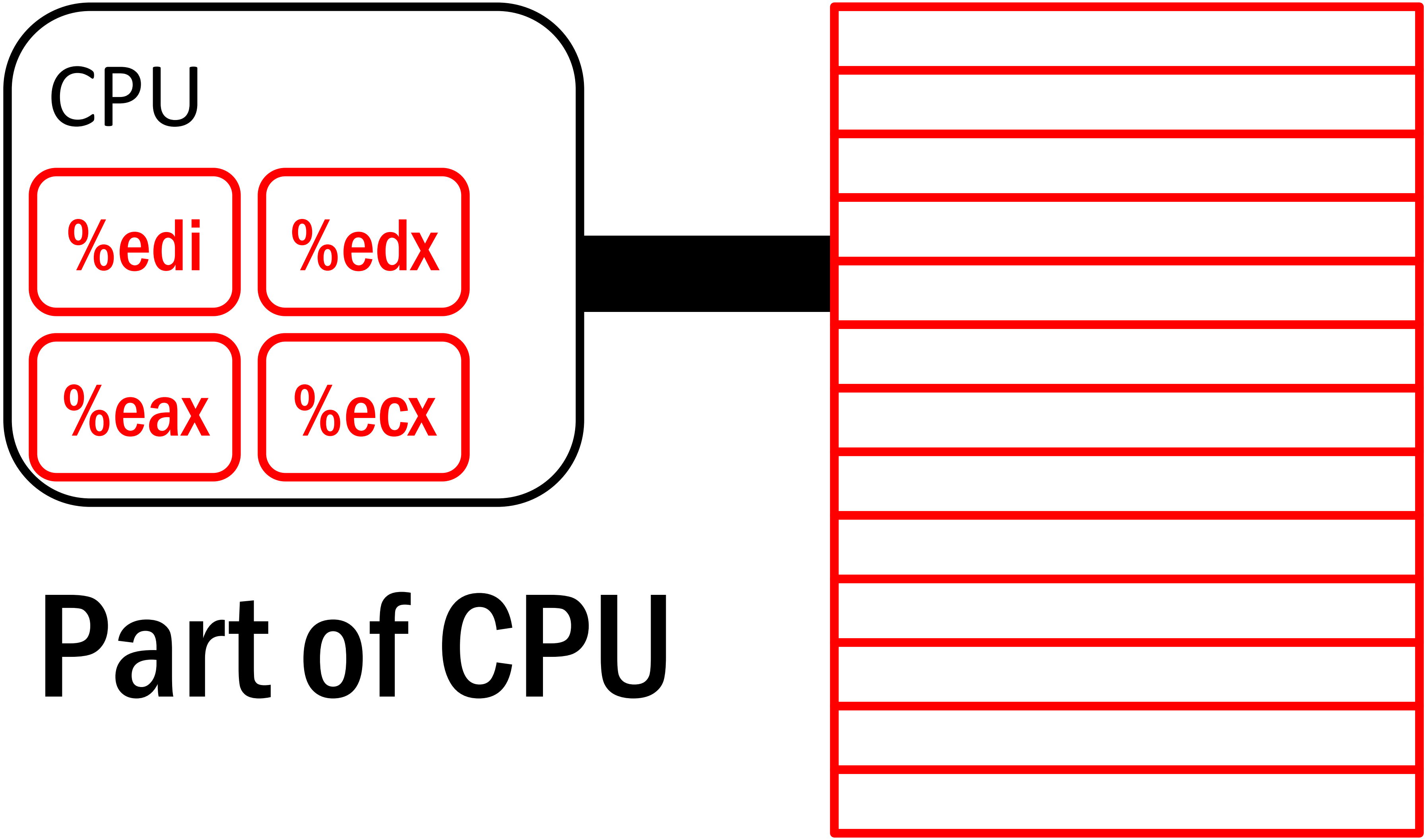


%edi %edx

%eax %ecx

Part of CPU

Over Memory Bus



CPU

%edi

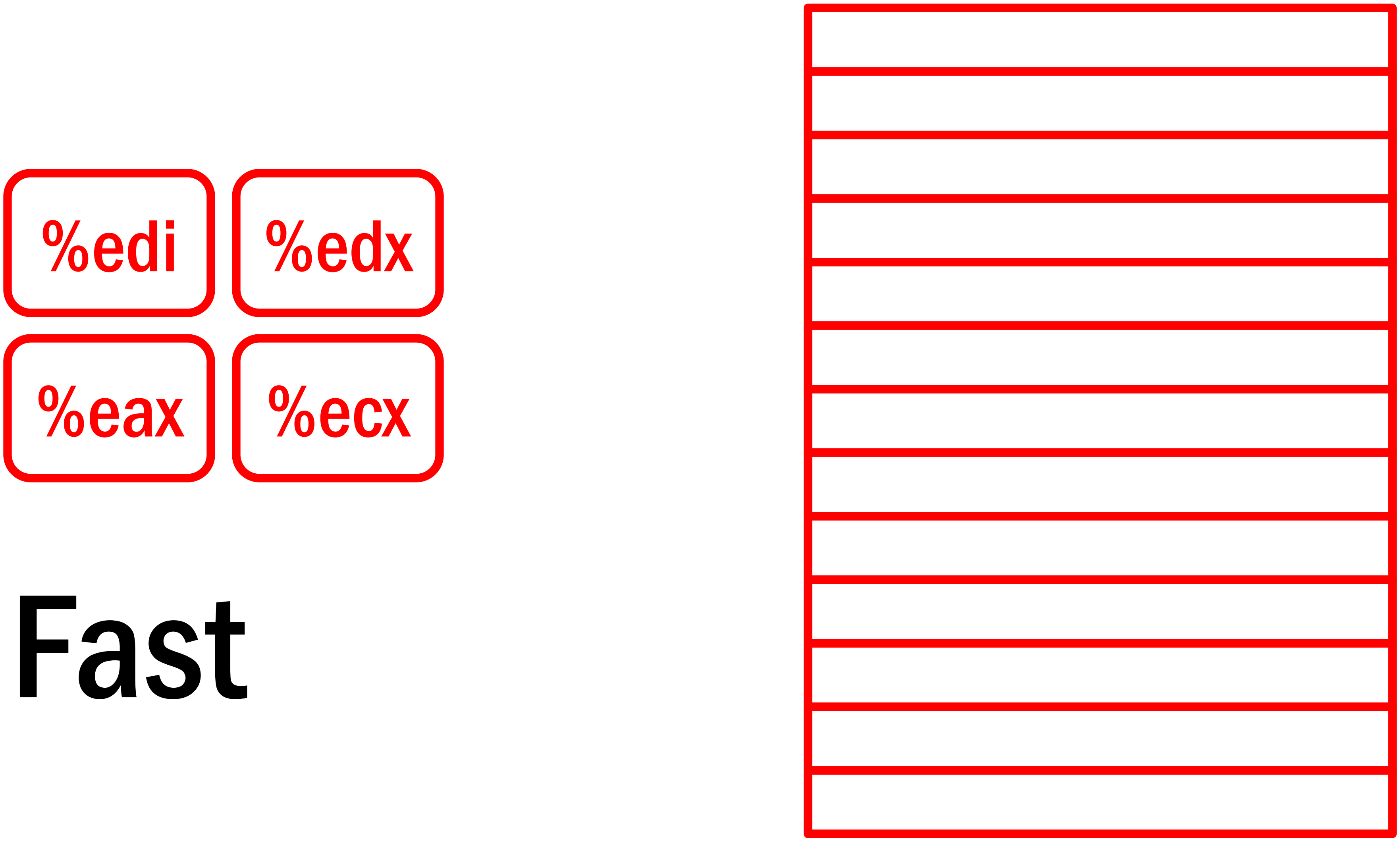
%edx

%eax

%ecx

Part of CPU

Over Memory Bus

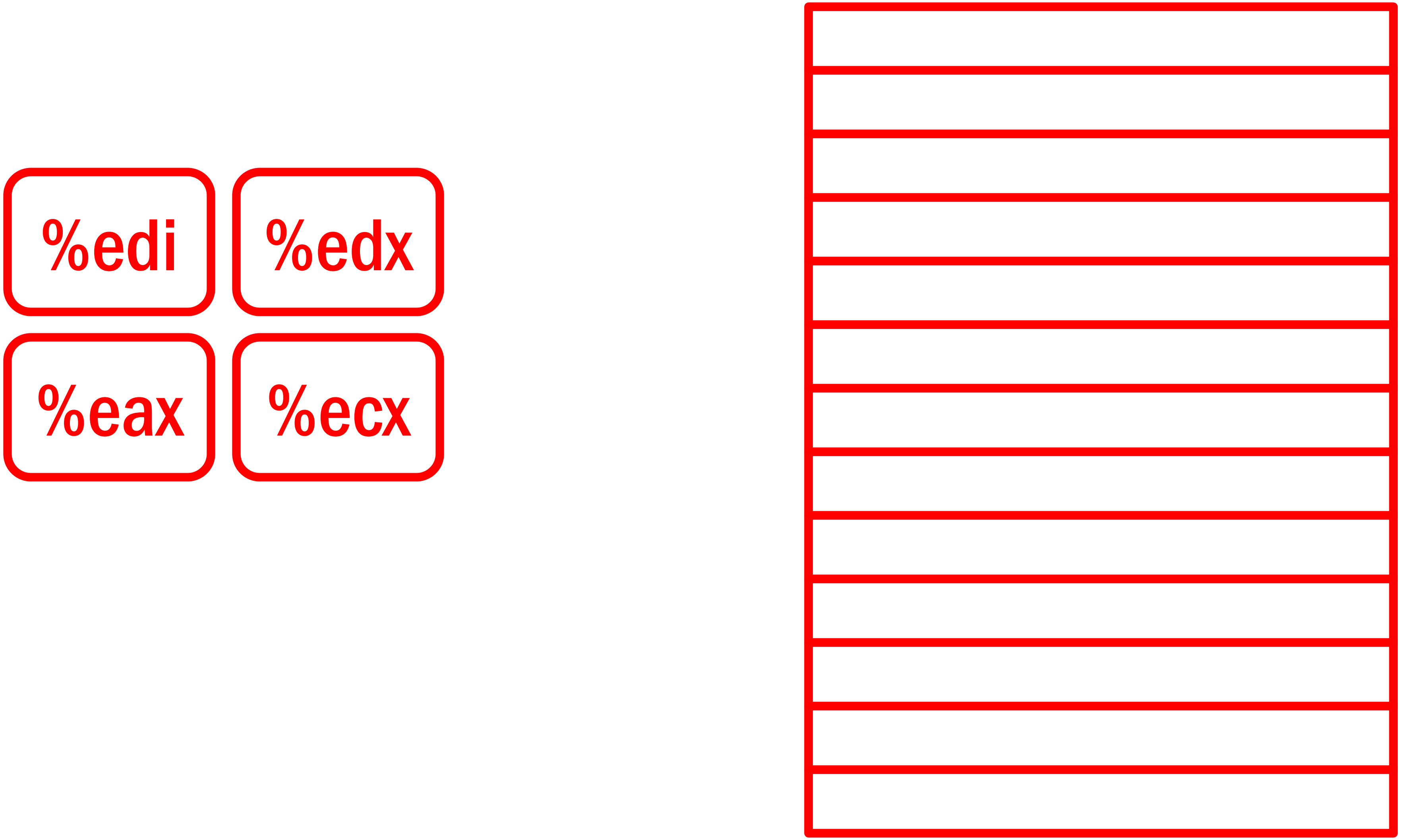


%edi %edx

%eax %ecx

Fast

Slow



%edi %edx

%eax %ecx

Idea: Use Registers instead of Memory

```
void func(char data[])
{
  int i;

  for (i=0; i<10; i++)
  {
    if (data[i] < ‘z’ && data[i] != ‘\n’)
      data[i]++;
  }
}
```

```
void func(char data[])
{
  int i;
  char c;

  for (i=0; i<10; i++)
  {
    c = data[i];

    if (c < ‘z’ && c != ‘\n’)
      data[i]++;
  }
}
```

Common Sub Expression

```
void func(int *x, int *y)
{
  int i;

  for (i=0; i<10; i++)
  {
    y[i] += *x;
  }
}
```

```c
void func(int *x, int *y)
{
  int i;
  int tmp = *x;

  for (i=0; i<10; i++)
  {
    y[i] += tmp;
  }
}
```

Why can’t a compiler do this?

```
void func(int *x, int *y)
{
  int i;
  int tmp = *x;

  for (i=0; i<10; i++)
  {
    y[i] += tmp;
  }
}
```

Why can’t a compiler do this? **Memory Aliasing**

```
void func(int a, int b, char data[])
{
  int i;

  for (i=0; i<10; i++)
  {
    data[a*b+i] = ‘A’;
  }
}
```

```
void func(int a, int b, char data[])
{
   int i;
   int tmp = a*b;

   for (i=0; i<10; i++)
   {
      data[tmp+i] = ‘A’;
   }
}
```

Code Hoisting

```
void func(int a, int b, char data[])
{
   int i;
   int tmp = a*b;
   data[tmp] = ‘A’;
   data[tmp+1] = ‘A’;
   data[tmp+2] = ‘A’;
   data[tmp+3] = ‘A’;
   data[tmp+4] = ‘A’;
   data[tmp+5] = ‘A’;
   data[tmp+6] = ‘A’;
   data[tmp+7] = ‘A’;
   data[tmp+8] = ‘A’;
   data[tmp+9] = ‘A’;
}
```

Loop Unrolling

Start Studying!