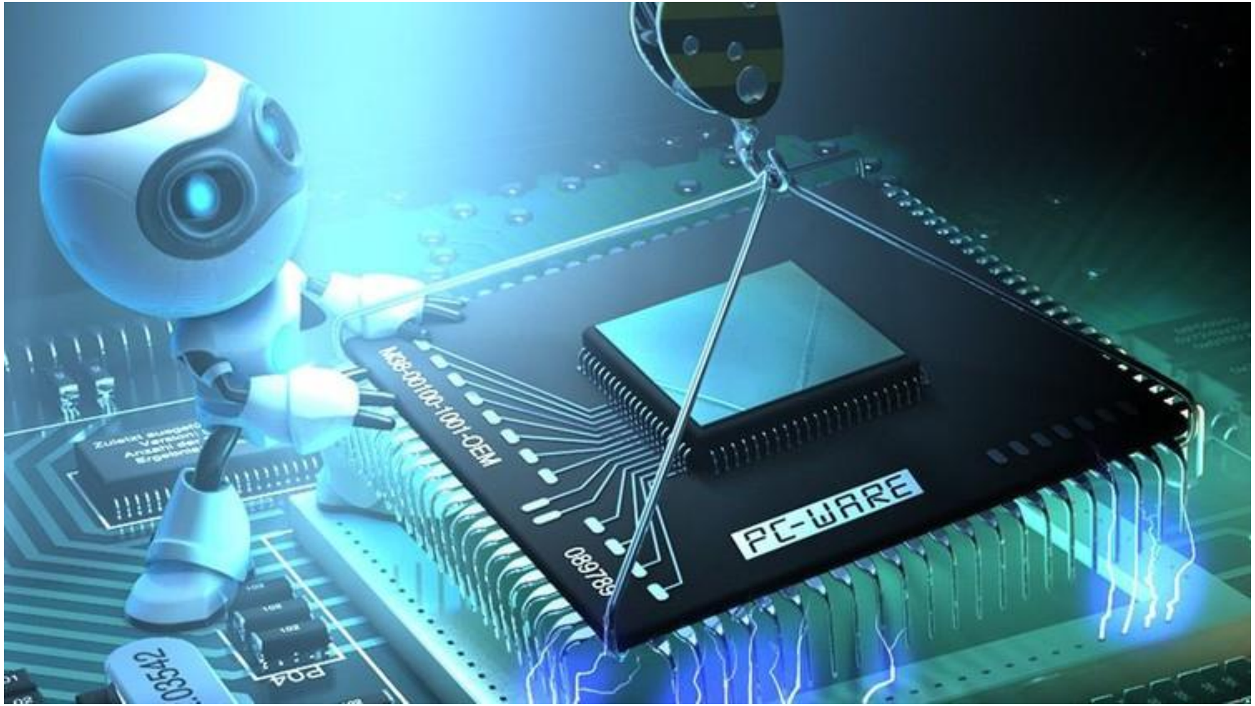
Working with Subroutines



Structured Data Types

Matrices

- **Declaration in C:**

  ```
  type name[NumRows][NumColumns];     // indexed starting at (0,0)
  ```

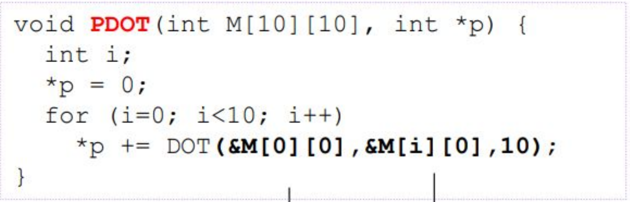
- **Storage by rows in consecutive memory locations**
  - Access element `A[i][j]`: **@start A + (i*NumColumns + j) * size**

    (size: size of the elements of A)

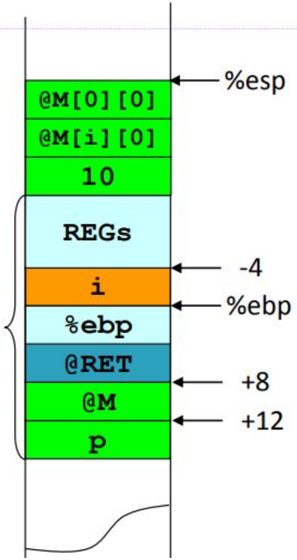
Working with Subroutines

1. Parameter passing

```
PDOT:   -
        -
        -
       pushl  $10
       imull  $10,-4(%ebp),%edx
       movl   8(%ebp),%ebx
       leal   (%ebx,%edx,4),%eax
       pushl  %eax
       pushl  %ebx
```



```
void PDOT(int M[10][10], int *p) {
  int i;
  *p = 0;
  for (i=0; i<10; i++)
    *p += DOT(&M[0][0],&M[i][0],10);
}
```



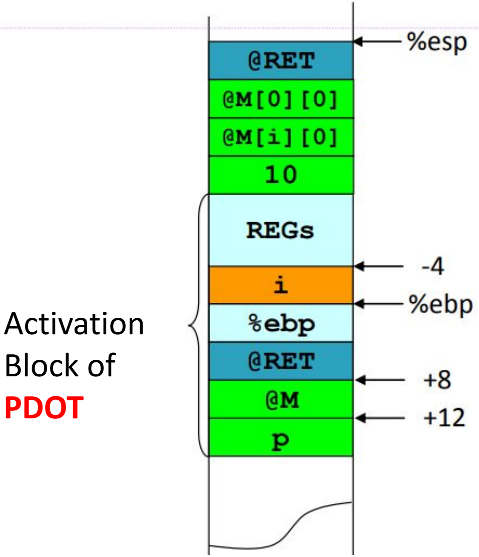
Activation Block of **PDOT**

Working with Subroutines

2. Subroutine call

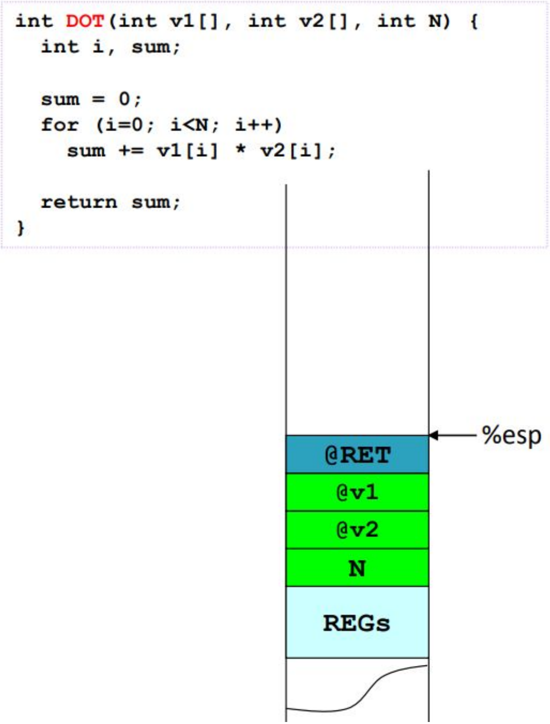
```
PDOT:   -
        -
        -
       pushl  $10
       imull  $10,-4(%ebp),%edx
       movl  8(%ebp),%ebx
       leal  (%ebx,%edx,4),%eax
       pushl  %eax
       pushl  %ebx
       call  DOT
```

```
void PDOT(int M[10][10], int *p) {
  int i;
  *p = 0;
  for (i=0; i<10; i++)
    *p += DOT(&M[0][0],&M[i][0],10);
}
```



Working with Subroutines

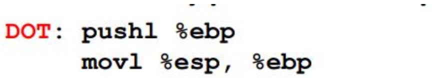
2. Subroutine call



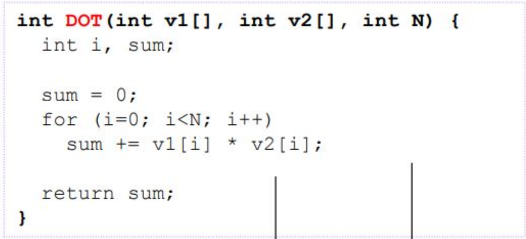
```
int DOT(int v1[], int v2[], int N) {
  int i, sum;

  sum = 0;
  for (i=0; i<N; i++)
    sum += v1[i] * v2[i];

  return sum;
}
```

Working with Subroutines

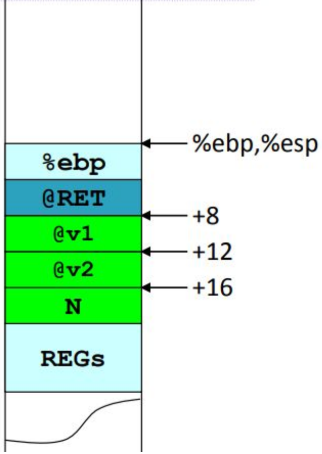
3. Dynamic link and pointer to activation block



```
DOT: pushl %ebp
     movl %esp, %ebp
```



```
int DOT(int v1[], int v2[], int N) {
  int i, sum;

  sum = 0;
  for (i=0; i<N; i++)
    sum += v1[i] * v2[i];

  return sum;
}
```

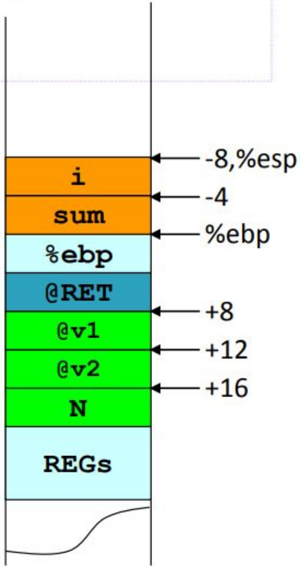


Working with Subroutines

4. Reserve space for local variables

```
DOT: pushl %ebp
     movl %esp, %ebp
     subl $8, %esp
```

```
int DOT(int v1[], int v2[], int N) {
  int i, sum;

  sum = 0;
  for (i=0; i<N; i++)
    sum += v1[i] * v2[i];

  return sum;
}
```

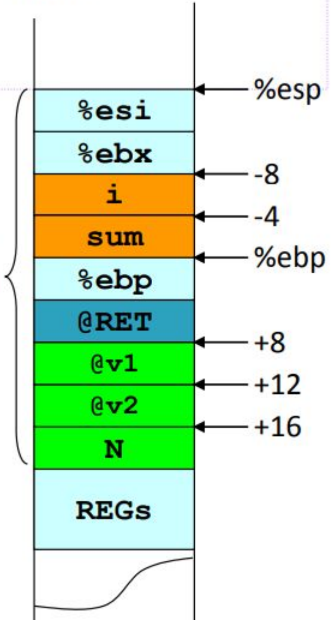


Working with Subroutines

5. Save caller state

```
DOT: pushl %ebp
     movl %esp, %ebp
     subl $8, %esp
     pushl %ebx
     pushl %esi
```

```
int DOT(int v1[], int v2[], int N) {
  int i, sum;

  sum = 0;
  for (i=0; i<N; i++)
    sum += v1[i] * v2[i];

  return sum;
}
```



Activation
Block of
DOT

Iterative Statement (FOR)

MODEL:

```
for (INI; COND; INC) {
   BODY-FOR
}
```

Generic translation:

```
       INI
for:   evaluate condition
       j(fails) end
       BODY-FOR
       INC
       jmp for
end:
```

Structured Data Types

Vectors

- **Declaration in C:**

```
type name[size];          // indexed starting at 0
```

- **Storage in consecutive memory locations**
  - ○ Access element `V[i]`: **@start V + i*size** (size: size of the elements of V)

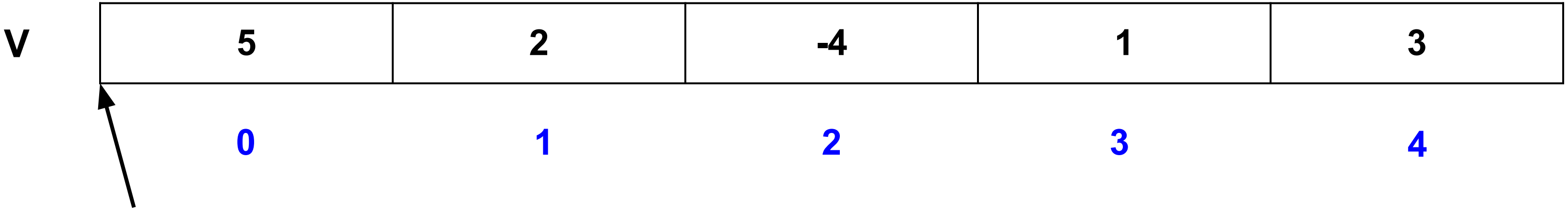
Structured Data Types

Vectors

- **Declaration in C:**

```
type name[size];          // indexed starting at 0
```

- **Storage in consecutive memory locations**
  - ○ Access element `V[i]`: **@start V + i*size** (size: size of the elements of V)



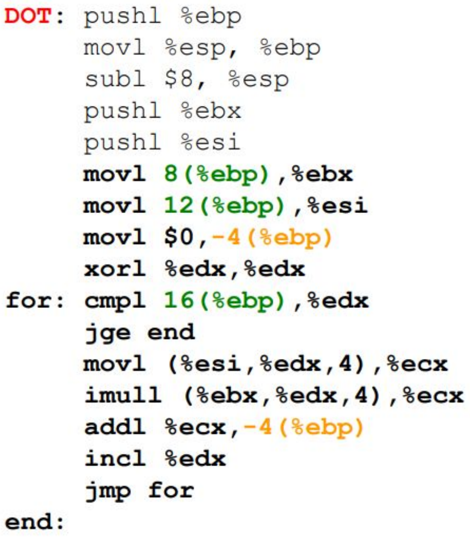
V	**5**	**2**	**-4**	**1**	**3**
	0	**1**	**2**	**3**	**4**

@V ≡ V[0]

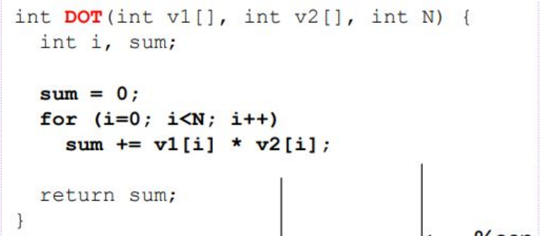
V[3] → @V + 3*4

Working with Subroutines

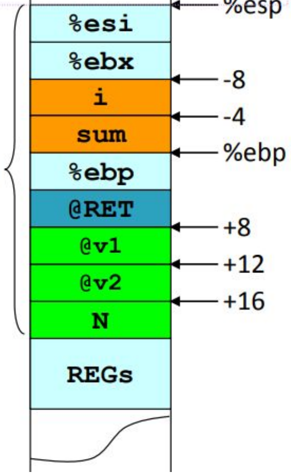
6. Subroutine Body



```
DOT: pushl %ebp
     movl %esp, %ebp
     subl $8, %esp
     pushl %ebx
     pushl %esi
     movl 8(%ebp),%ebx
     movl 12(%ebp),%esi
     movl $0,-4(%ebp)
     xorl %edx,%edx
for: cmpl 16(%ebp),%edx
     jge end
     movl (%esi,%edx,4),%ecx
     imull (%ebx,%edx,4),%ecx
     addl %ecx,-4(%ebp)
     incl %edx
     jmp for
end:
```



```
int DOT(int v1[], int v2[], int N) {
  int i, sum;

  sum = 0;
  for (i=0; i<N; i++)
    sum += v1[i] * v2[i];

  return sum;
}
```



Activation
Block of
DOT

Lucas Bazilio - Udemy

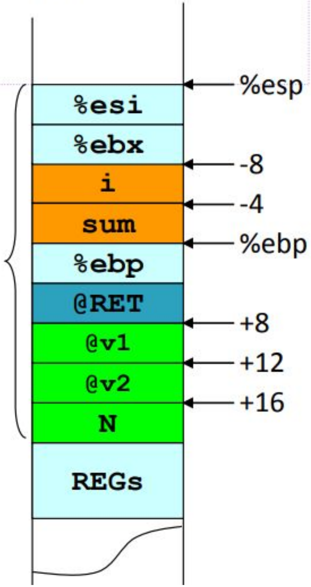
Working with Subroutines

7. Move result to %eax

```
DOT: pushl %ebp
     movl %esp, %ebp
     subl $8, %esp
     pushl %ebx
     pushl %esi
     movl 8(%ebp),%ebx
     movl 12(%ebp),%esi
     movl $0,-4(%ebp)
     xorl %edx,%edx
for: cmpl 16(%ebp),%edx
     jge end
     movl (%esi,%edx,4),%ecx
     imull (%ebx,%edx,4),%ecx
     addl %ecx,-4(%ebp)
     incl %edx
     jmp for
end: movl -4(%ebp),%eax
```

```
int DOT(int v1[], int v2[], int N) {
  int i, sum;

  sum = 0;
  for (i=0; i<N; i++)
    sum += v1[i] * v2[i];

  return sum;
}
```

Activation
Block of
DOT

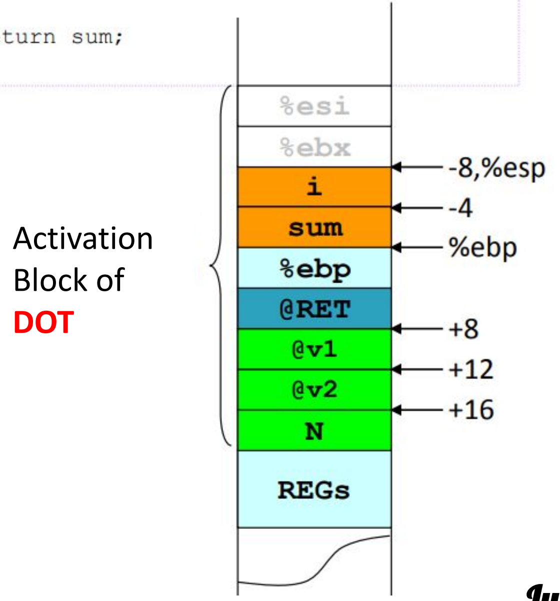


Working with Subroutines

8. Restore caller state

```
DOT:  pushl %ebp
      movl %esp, %ebp
      subl $8, %esp
      pushl %ebx
      pushl %esi
      movl 8(%ebp),%ebx
      movl 12(%ebp),%esi
      movl $0,-4(%ebp)
      xorl %edx,%edx
for:  cmpl 16(%ebp),%edx
      jge end
      movl (%esi,%edx,4),%ecx
      imull (%ebx,%edx,4),%ecx
      addl %ecx,-4(%ebp)
      incl %edx
      jmp for
end:  movl -4(%ebp),%eax
      popl %esi
      popl %ebx
```

```
int DOT(int v1[], int v2[], int N) {
  int i, sum;

  sum = 0;
  for (i=0; i<N; i++)
    sum += v1[i] * v2[i];

  return sum;
}
```

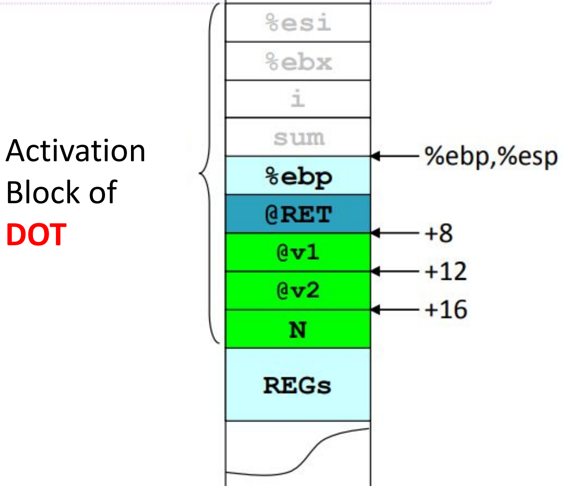


Working with Subroutines

9. Remove local variables

```
DOT: pushl %ebp
     movl %esp, %ebp
     subl $8, %esp
     pushl %ebx
     pushl %esi
     movl 8(%ebp),%ebx
     movl 12(%ebp),%esi
     movl $0,-4(%ebp)
     xorl %edx,%edx
for: cmpl 16(%ebp),%edx
     jge end
     movl (%esi,%edx,4),%ecx
     imull (%ebx,%edx,4),%ecx
     addl %ecx,-4(%ebp)
     incl %edx
     jmp for
end: movl -4(%ebp),%eax
     popl %esi
     popl %ebx
     movl %ebp,%esp
```

```
int DOT(int v1[], int v2[], int N) {
  int i, sum;

  sum = 0;
  for (i=0; i<N; i++)
    sum += v1[i] * v2[i];

  return sum;
}
```

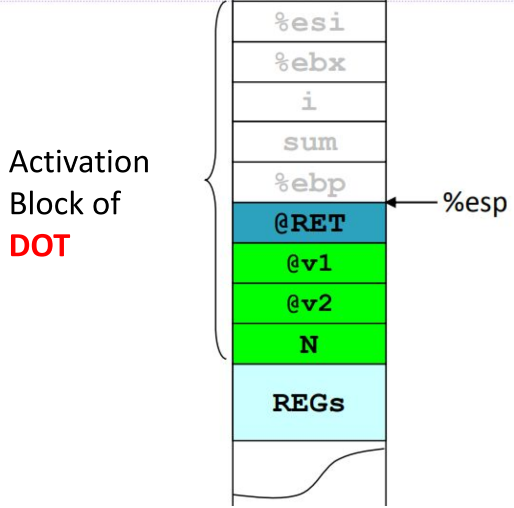


Working with Subroutines

10. Undo dynamic link

```
DOT: pushl %ebp
     movl %esp, %ebp
     subl $8, %esp
     pushl %ebx
     pushl %esi
     movl 8(%ebp),%ebx
     movl 12(%ebp),%esi
     movl $0,-4(%ebp)
     xorl %edx,%edx
for: cmpl 16(%ebp),%edx
     jge end
     movl (%esi,%edx,4),%ecx
     imull (%ebx,%edx,4),%ecx
     addl %ecx,-4(%ebp)
     incl %edx
     jmp for
end: movl -4(%ebp),%eax
     popl %esi
     popl %ebx
     movl %ebp,%esp
     popl %ebp
```

```
int DOT(int v1[], int v2[], int N) {
  int i, sum;

  sum = 0;
  for (i=0; i<N; i++)
    sum += v1[i] * v2[i];

  return sum;
}
```



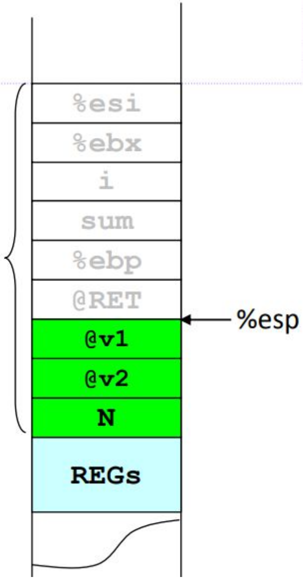
Working with Subroutines

11. Return subroutine

```
DOT:  pushl %ebp
      movl %esp, %ebp
      subl $8, %esp
      pushl %ebx
      pushl %esi
      movl 8(%ebp),%ebx
      movl 12(%ebp),%esi
      movl $0,-4(%ebp)
      xorl %edx,%edx
for:  cmpl 16(%ebp),%edx
      jge end
      movl (%esi,%edx,4),%ecx
      imull (%ebx,%edx,4),%ecx
      addl %ecx,-4(%ebp)
      incl %edx
      jmp for
end:  movl -4(%ebp),%eax
      popl %esi
      popl %ebx
      movl %ebp,%esp
      popl %ebp
      ret
```

```
int DOT(int v1[], int v2[], int N) {
  int i, sum;

  sum = 0;
  for (i=0; i<N; i++)
    sum += v1[i] * v2[i];

  return sum;
}
```

Activation Block of DOT

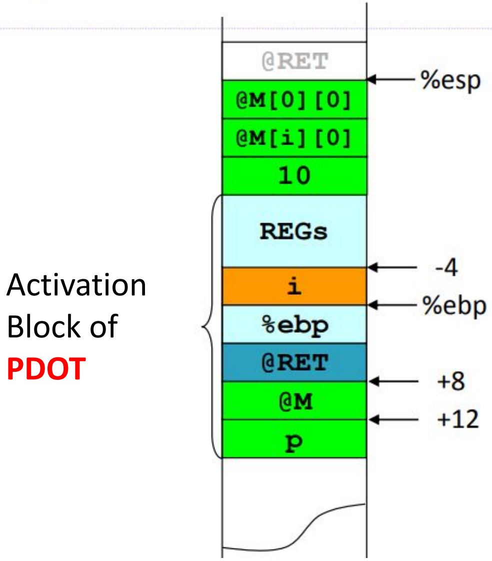


Working with Subroutines

11. We return to the subroutine

```
void PDOT(int M[10][10], int *p) {
  int i;
  *p = 0;
  for (i=0; i<10; i++)
    *p += DOT(&M[0][0],&M[i][0],10);
}
```

```
PDOT:  -
       -
       -
       pushl $10
       imull $10,-4(%ebp),%edx
       movl 8(%ebp),%ebx
       leal (%ebx,%edx,4),%eax
       pushl %eax
       pushl %ebx
       call DOT
```

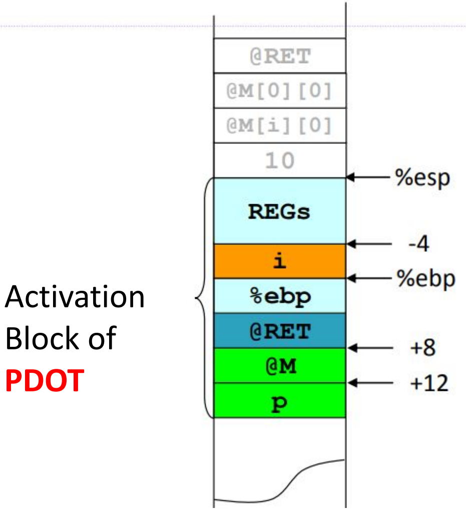


Working with Subroutines

12. Remove parameters

```
void PDOT(int M[10][10], int *p) {
  int i;
  *p = 0;
  for (i=0; i<10; i++)
    *p += DOT(&M[0][0],&M[i][0],10);
}
```

```
PDOT:   -
        -
        -
       pushl  $10
       imull  $10,-4(%ebp),%edx
       movl  8(%ebp),%ebx
       leal  (%ebx,%edx,4),%eax
       pushl  %eax
       pushl  %ebx
       call  DOT
       addl  $12,%esp
```



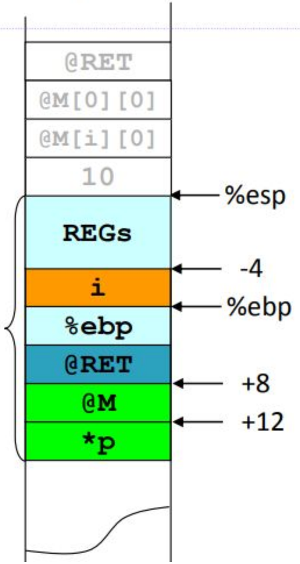
Working with Subroutines

13. Collect/use result

```
void PDOT(int M[10][10], int *p) {
  int i;
  *p = 0;
  for (i=0; i<10; i++)
    *p += DOT(&M[0][0],&M[i][0],10);
}
```

```
PDOT:   -
        -
        -
       pushl $10
       imull $10,-4(%ebp),%edx
       movl  8(%ebp),%ebx
       leal  (%ebx,%edx,4),%eax
       pushl %eax
       pushl %ebx
       call  DOT
       addl  $12,%esp
       movl  12(%ebp),%ebx
       addl  %eax,(%ebx)
```

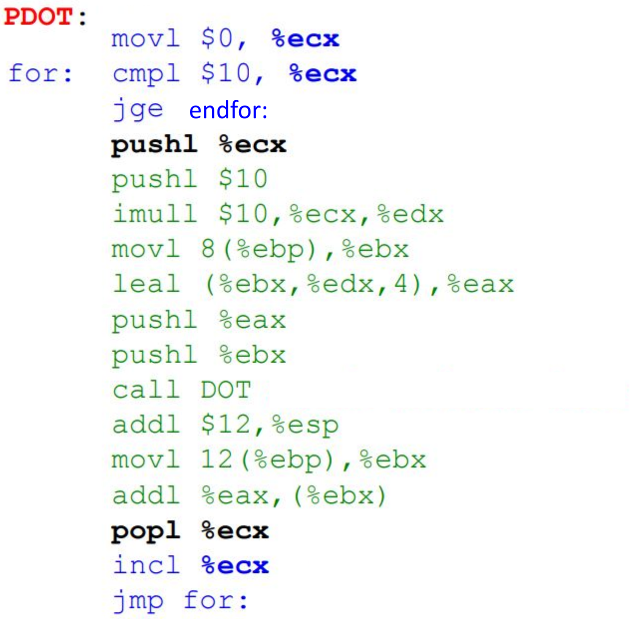
Activation Block of PDOT



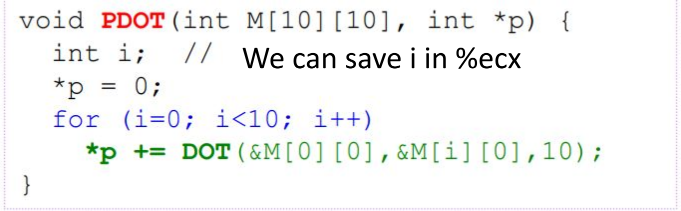
Lucas Bazilio - Udemy

Working with Subroutines

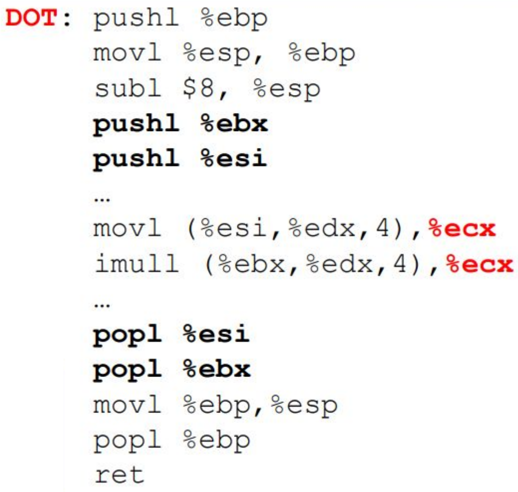
Using the for loop



```
PDOT:
        movl $0, %ecx
 for:   cmpl $10, %ecx
        jge  endfor:
        pushl %ecx
        pushl $10
        imull $10,%ecx,%edx
        movl 8(%ebp),%ebx
        leal (%ebx,%edx,4),%eax
        pushl %eax
        pushl %ebx
        call DOT
        addl $12,%esp
        movl 12(%ebp),%ebx
        addl %eax,(%ebx)
        popl %ecx
        incl %ecx
        jmp for:
endfor:
```



```
void PDOT(int M[10][10], int *p) {
  int i;  //  We can save i in %ecx
  *p = 0;
  for (i=0; i<10; i++)
    *p += DOT(&M[0][0],&M[i][0],10);
}
```



```
DOT:  pushl %ebp
      movl %esp, %ebp
      subl $8, %esp
      pushl %ebx
      pushl %esi
      …
      movl (%esi,%edx,4),%ecx
      imull (%ebx,%edx,4),%ecx
      …
      popl %esi
      popl %ebx
      movl %ebp,%esp
      popl %ebp
      ret
```