



Runtime Environment

Dr. D. M. Akbar Hussain
Department of Electronic Systems

1



Runtime Environment

Basically, related to the hardware structure of registers and memory of a machine which serves as a tool for managing information for execution.

- Fully Static Environment
FORTRAN
- Stack Based Environment
C, C++, Pascal and Ada
- Fully Dynamic Environment
LISP

Dr. D. M. Akbar Hussain
Department of Electronic Systems

2



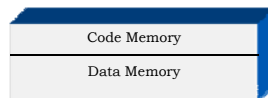
Memory Division

➤ Memory is sub-divided conceptually into **Code** and **Data** area

- ❑ **Code** area addresses are computed at compile time.
- ❑ Same is not true for data area.

Addresses can be computer for global and static data.

An Exception: Global/static data related to const, strings are typically inserted into the code by the compiler, same is true for global functions as there entry points are known.

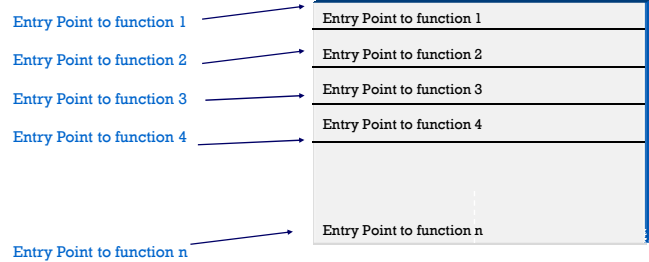


Dr. D. M. Akbar Hussain
Department of Electronic Systems

3



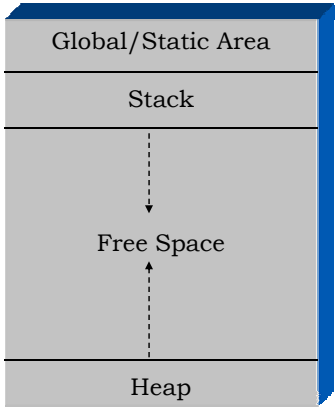
Code Memory




Dr. D. M. Akbar Hussain
Department of Electronic Systems

4

Data Memory



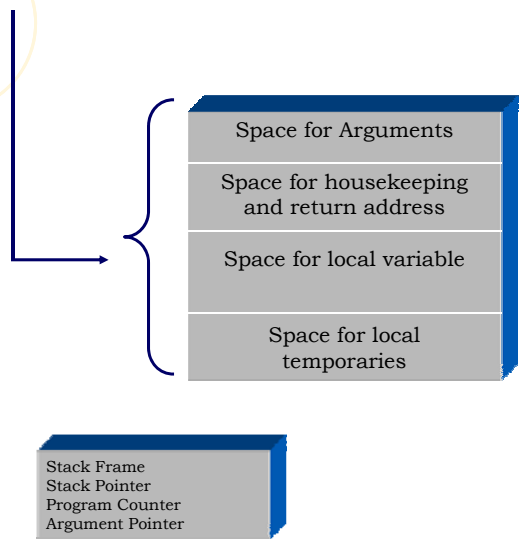
The diagram illustrates the layout of data memory as a vertical stack of four sections. From top to bottom, they are: Global/Static Area, Stack, Free Space, and Heap. A dashed arrow points downwards from the Stack section to the Free Space section, and another dashed arrow points upwards from the Heap section to the Free Space section, indicating that the Stack grows downwards and the Heap grows upwards.




Dr. D. M. Akbar Hussain
Department of Electronic Systems

5

Activation Record




The diagram shows the structure of an activation record. It is represented as a vertical stack of four sections: Space for Arguments, Space for housekeeping and return address, Space for local variable, and Space for local temporaries. A bracket on the left groups these four sections together. Below this stack, a separate box lists the pointers associated with the activation record: Stack Frame, Stack Pointer, Program Counter, and Argument Pointer. An arrow points from the top of the activation record stack to the Stack Pointer entry in the pointer list.



Dr. D. M. Akbar Hussain
Department of Electronic Systems

6

Calling & Return Operation




Determination of sequence of operations resulting from a call (procedure/function)
Issues: Division of responsibility between caller and callee and how much to rely on processor
As it is easy to call from the caller side but which also means duplicating the same code every time a call is made (space issue)
At a minimum caller must compute the arguments and place them in callees record or some where callee can find it.
Machine registers are used at the time of call so status and all necessary information has to be stored for later use when returning from call. Who should do it: Probably both (Caller & Callee).

Dr. D. M. Akbar Hussain
Department of Electronic Systems

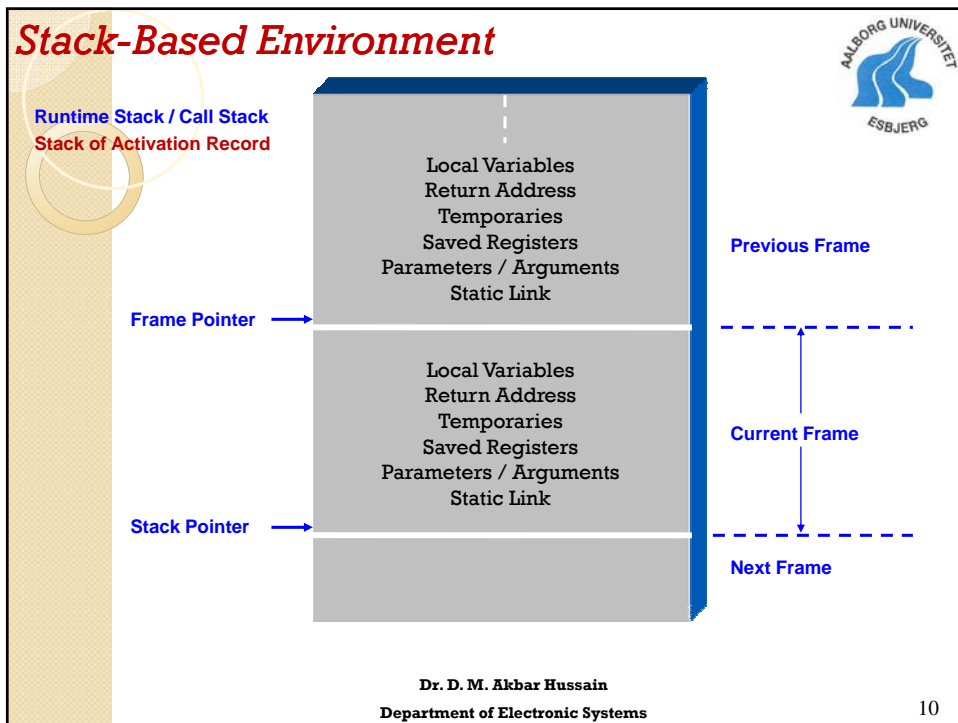
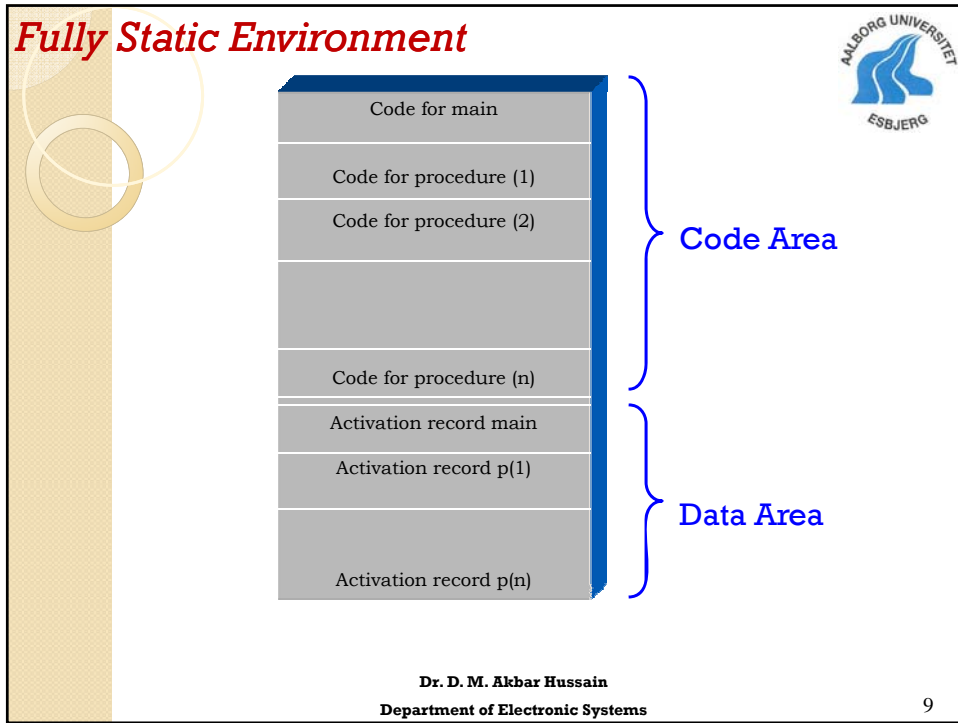
7

Responsibility Division




Dr. D. M. Akbar Hussain
Department of Electronic Systems

8



Stack-Based Environment



Stack-Based without Local Procedures


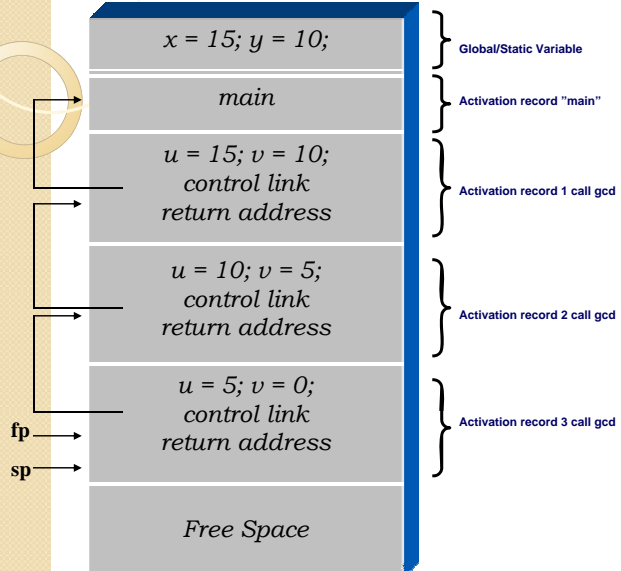
```

#include <stdio.h>
int x, y;
int gcd (int u, int v)
{ if (v == 0) return u;
  else return gcd (v, u%v); }
main()
{
scanf ("%d %d", &x, &y);
printf ("%d \n", gcd (x, y));
return 0;
}
    
```

Dr. D. M. Akbar Hussain
Department of Electronic Systems

11

Stack Layout

```

#include <stdio.h>
int x, y;
int gcd (int u, int v)
{ if (v == 0) return u;
  else return gcd (v, u%v); }
main()
{
scanf ("%d %d", &x, &y);
printf ("%d \n", gcd (x, y));
return 0;
}
    
```

Dr. D. M. Akbar Hussain
Department of Electronic Systems

12



Example

```
#include <stdio.h>
int x = 2; void g (int n); /* prototype */

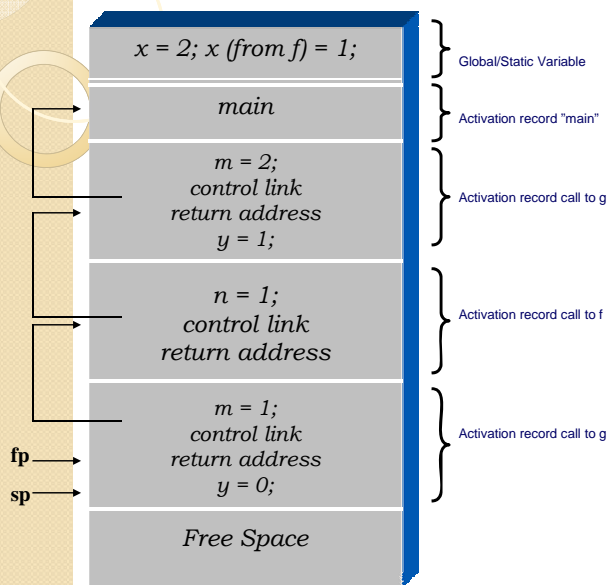
void f (int n)
{ static int x = 1;
  g (n); x--;}

void g (int m)
{ int y = m - 1;
  if (y > 0)
    { f (y); x--; g (y); }
}

main()
{
  g (x);
  return 0;
}
```

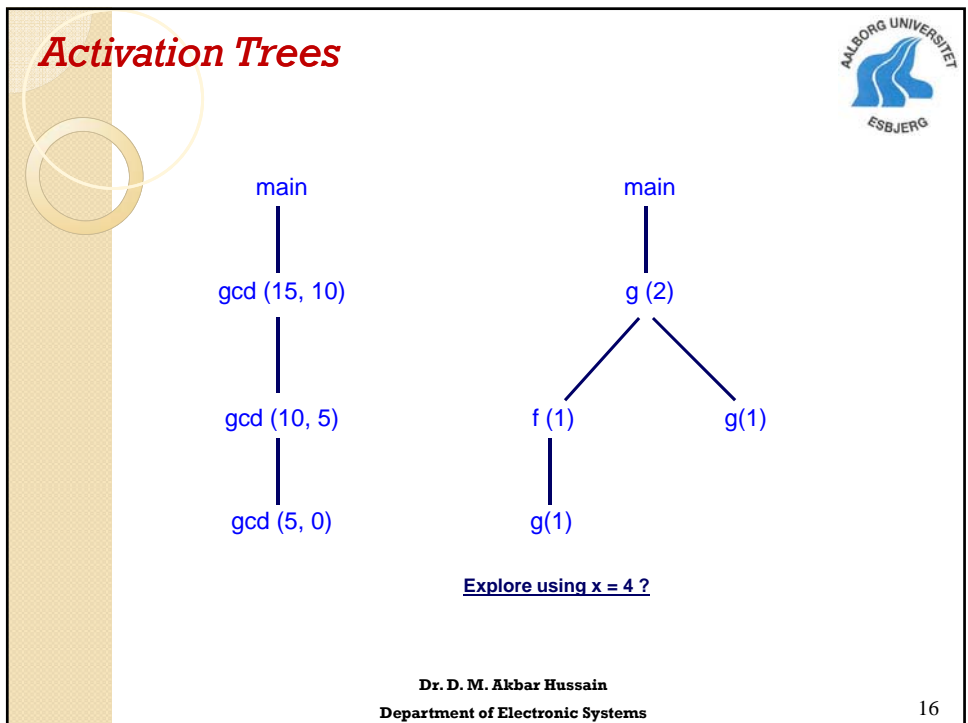
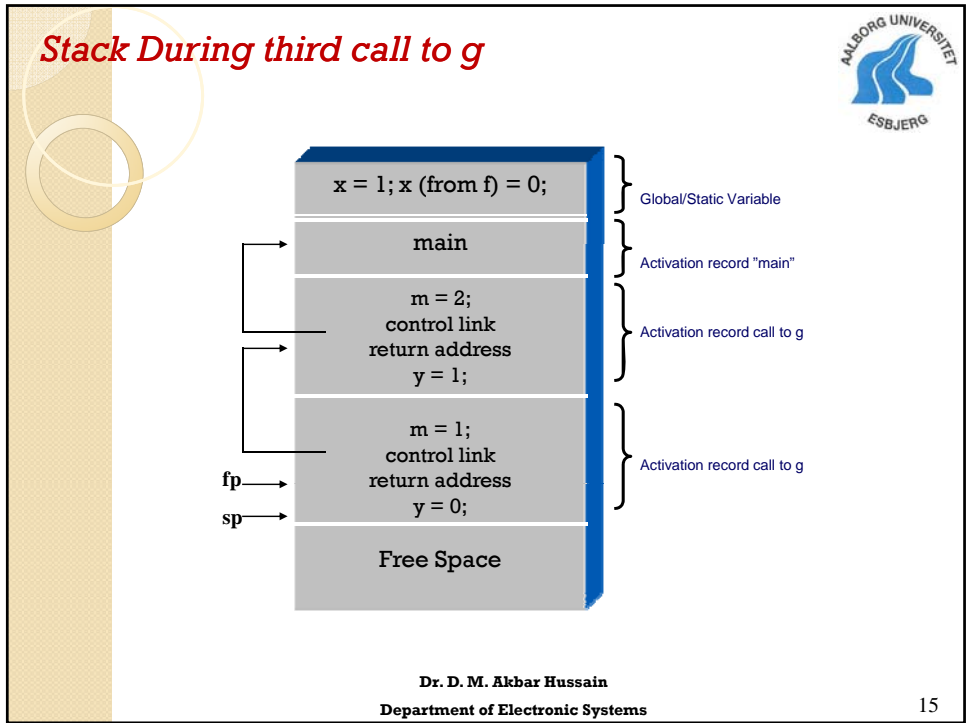
Dr. D. M. Akbar Hussain
Department of Electronic Systems



Stack During Second call to g



```
#include <stdio.h>
int x = 2;
void g (int n); /* prototype */
void f (int n)
{ static int x = 1;
  g (n); x--;}
void g (int m)
{ int y = m - 1;
  if (y > 0)
    { f (y); x--; g (y); }
}
main()
{
  g (x);
  return 0;
}
```

Dr. D. M. Akbar Hussain
Department of Electronic Systems





Calling/Return Sequence

- Compute the arguments and push them in the new activation record.
- Push the fp as the control link in the new activation record.
- Change the fp so that it points to the beginning of the new activation record (copy sp into fp).
- Store the return address in the new activation record.
- Jump to the code of the procedure to be called.

- Copy fp to sp
- Load control link into fp
- Jump to the return address
- Change the sp for popping

Dr. D. M. Akbar Hussain
Department of Electronic Systems

17