Al-Mustaqbal University
College of Sciences
Intelligent Medical System Department

Embedded systems

Lecture 6 :

Memory II

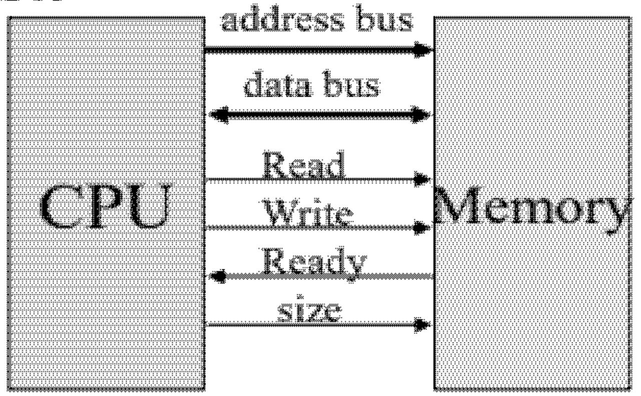
Prof.Dr. Mehdi Ebady Manaa

جامعة المستقبل
AL MUSTAQBAL UNIVERSITY

CPU Memory Interface

CPU Memory Interface usually consists of:

- unidirectional address bus
- bidirectional data bus
- read control line
- write control line
- ready control line
- size (byte, word) control line



Memory access involves a memory bus transaction

- read:

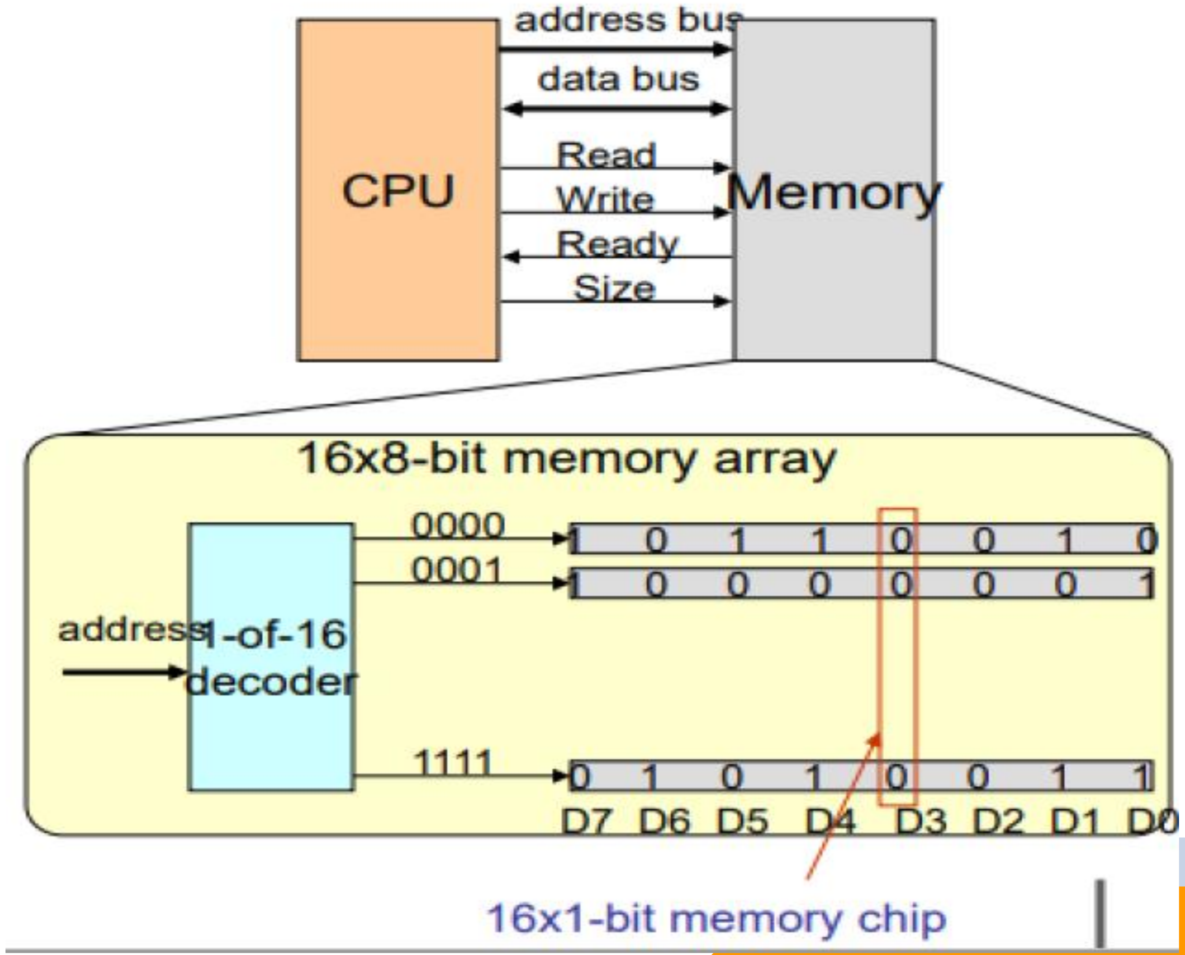
  (1) set address, read and size,
  (2) copy data when ready is set by memory

- **write:**

  (1) set address, data, write and size,
  (2) done when ready is set

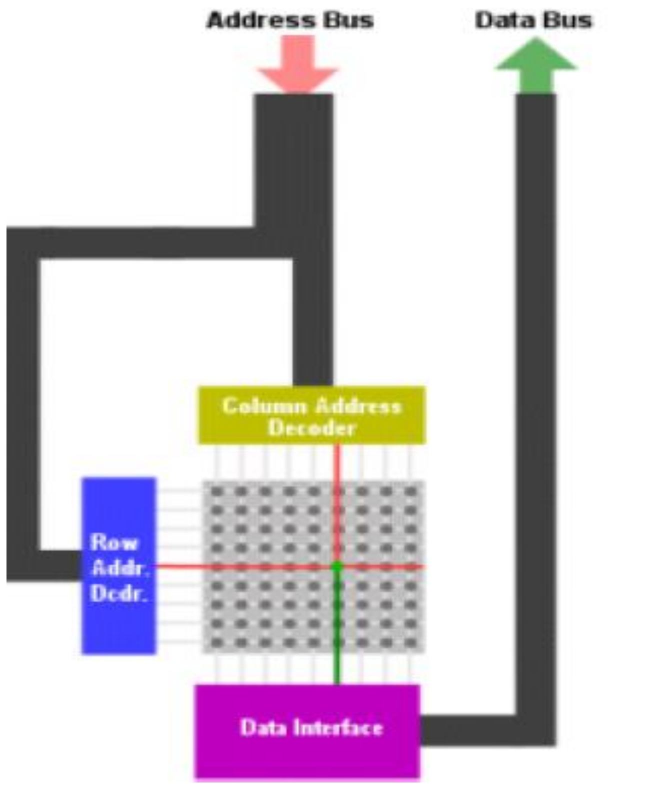
CPU Memory Interface

- Memory subsystems generally consist of chips+controller
- Each chip provides few bits (e.g., 14) per access
  - Bits from multiple chips are accessed in parallel to fetch bytes and words
  - Memory controller decodes/translates address and control signals
  - Controller can also be on memory chip
- Example:
  - contains 8 *16x1*bit chips and very simple controller

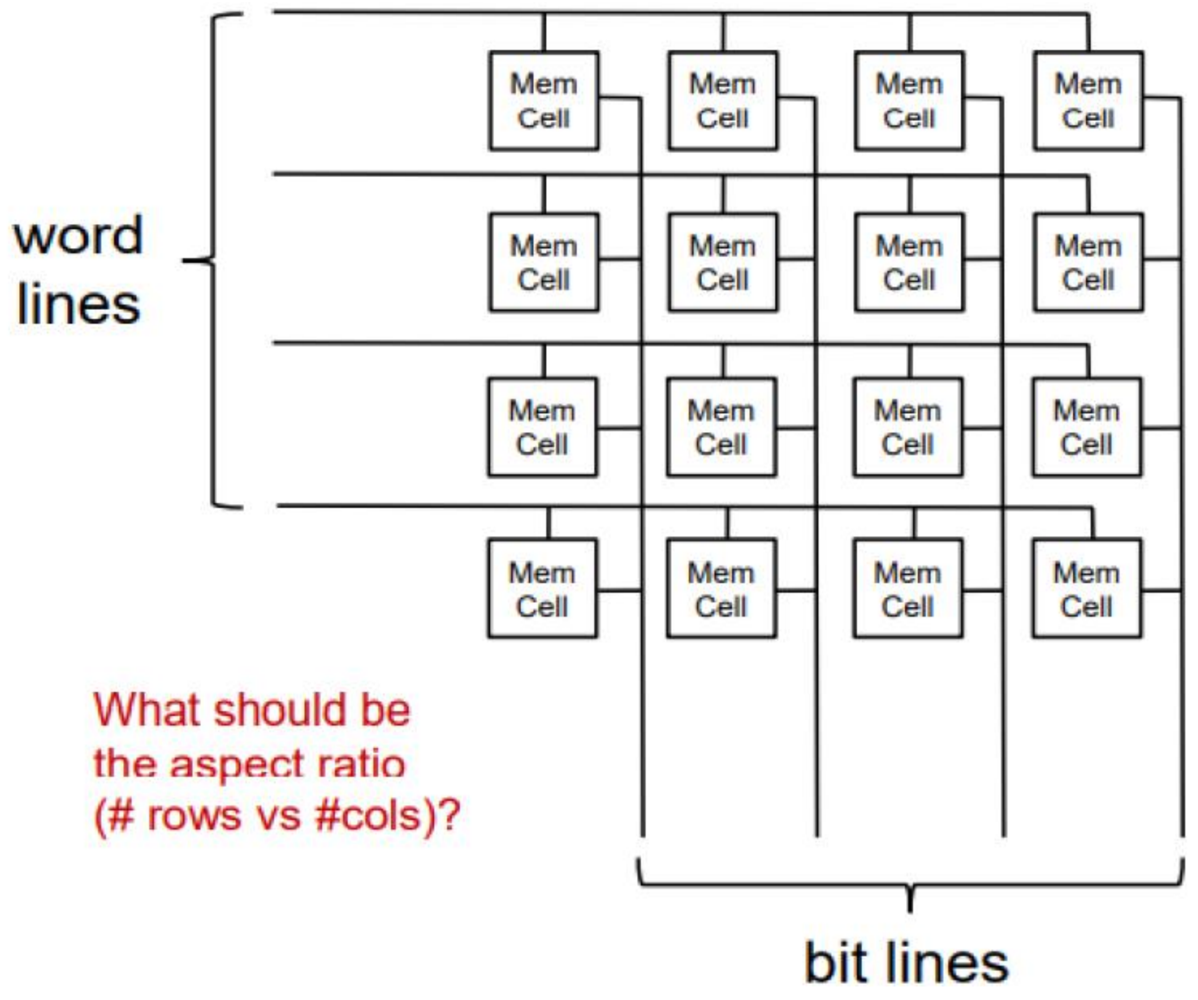


Storage Basics

- Just because the CPU sees RAM as one long, thin line of bytes doesn't mean that it's actually laidout that way.
- Real RAM chips don't store whole bytes, but rather they storeindividual bits in a grid, which you can address one bit at a time.
- Types of memory.
- **Non Volatile >ROM/EPROM/FLASH.**
- **Volatile > SRAM, DRAM**



Internal organization

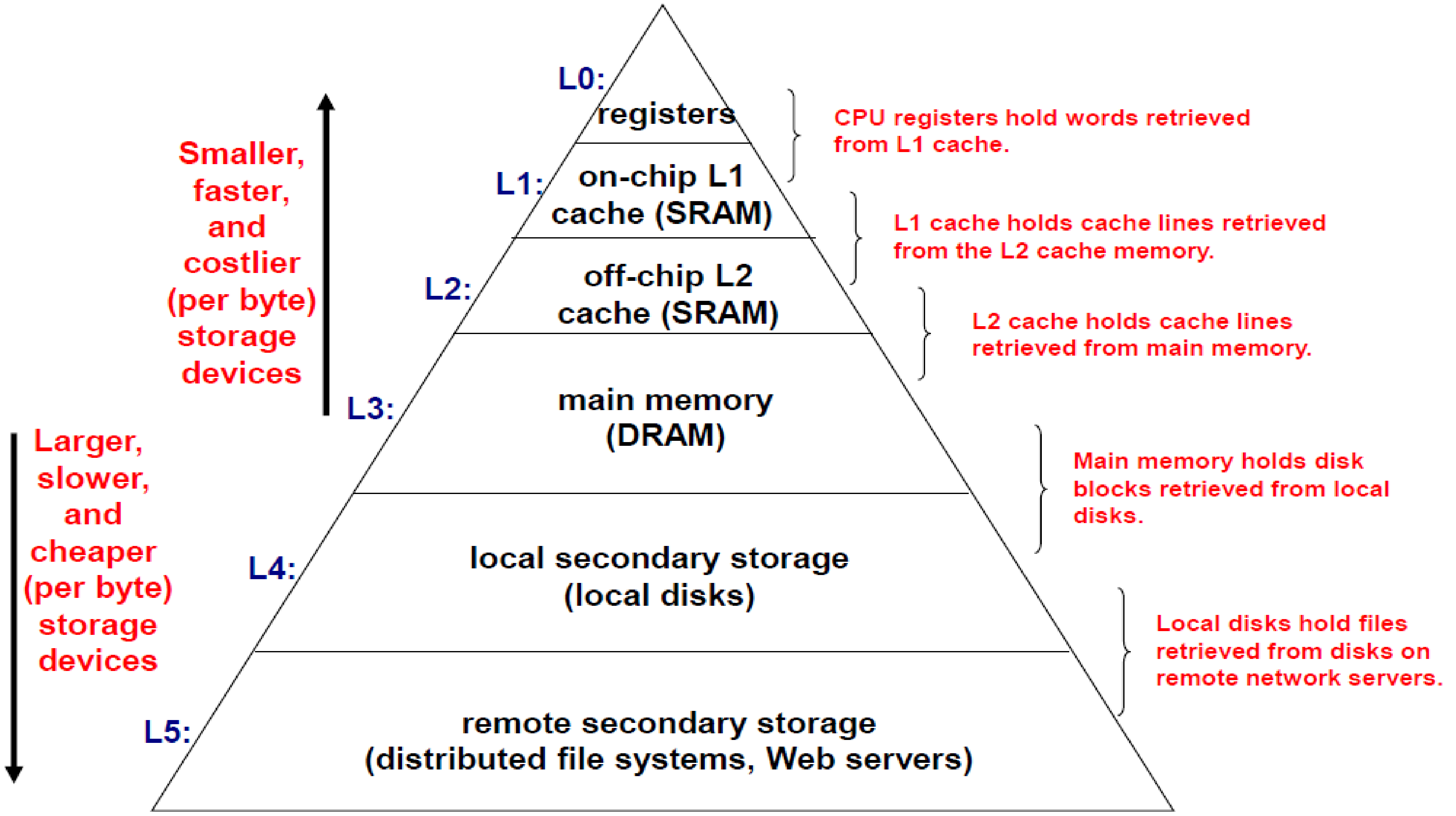


What should be the aspect ratio (# rows vs #cols)?

Different memory types (e.g. SRAM vs DRAM) are distinguished by the technology used to implement the memory cell, e.g.:

- SRAM: 6T
- DRAM: 1T/1C

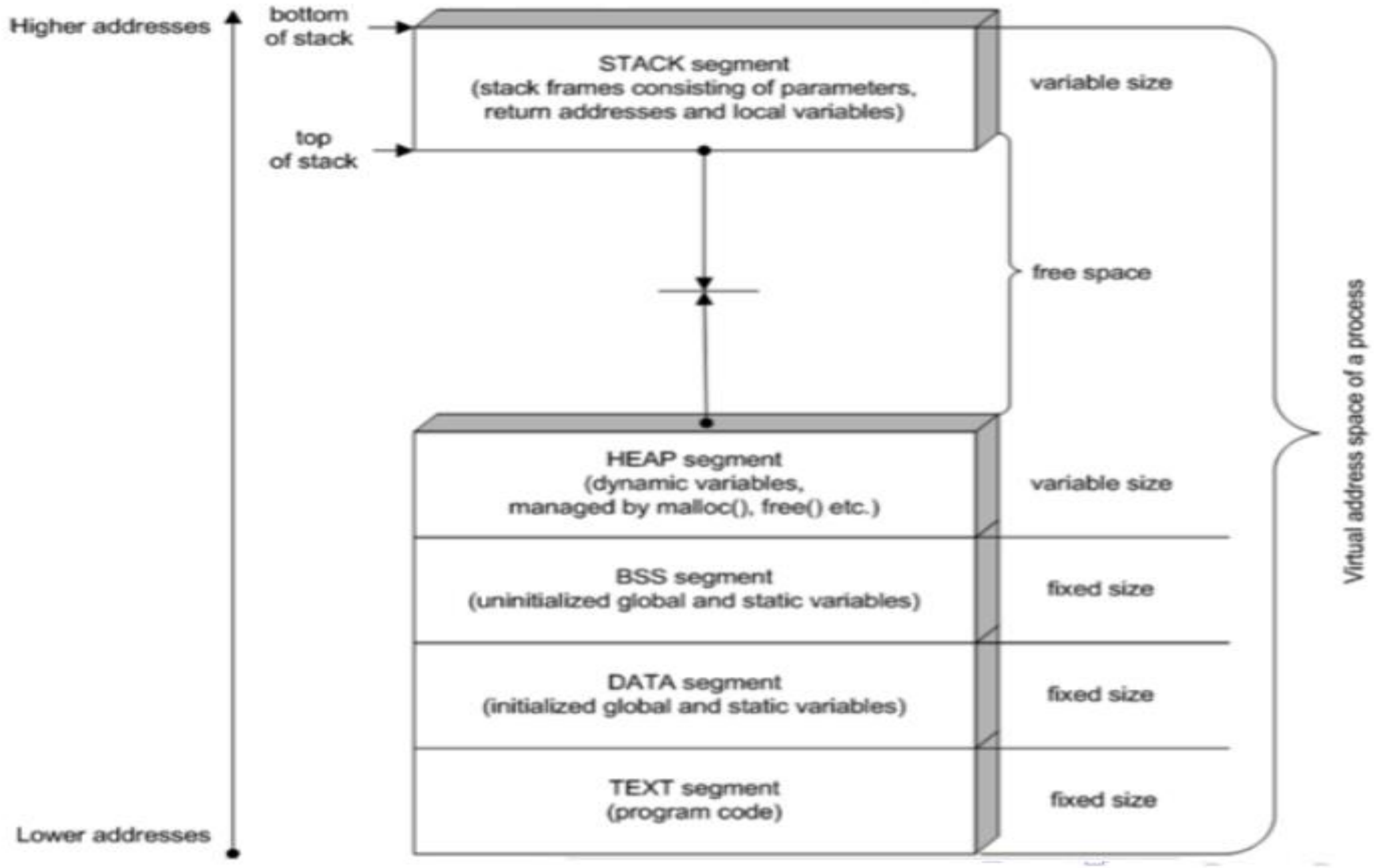
An Example Memory Hierarchy


L0:
registers
L1:
on-chip L1 cache (SRAM)
L2:
off-chip L2 cache (SRAM)
L3:
main memory (DRAM)
L4:
local secondary storage (local disks)
L5:
remote secondary storage (distributed file systems, Web servers)
Smaller, faster, and costlier (per byte) storage devices
Larger, slower, and cheaper (per byte) storage devices
CPU registers hold words retrieved from L1 cache.
L1 cache holds cache lines retrieved from the L2 cache memory.
L2 cache holds cache lines retrieved from main memory.
Main memory holds disk blocks retrieved from local disks.
Local disks hold files retrieved from disks on remote network servers.

Local, Global and Static memory

- A local variable is one that occurs within a specific scope. They exist only in the function where they are created.
- A global variable is a variable that is defined outside all functions and available to all functions.
- In local variables, static is used to store the variable in the statically allocated memory instead of the automatically allocated memory.
- Statically allocated memory (global or static) is typically reserved in
- data segment of the program at compile time.

Program Memory Map

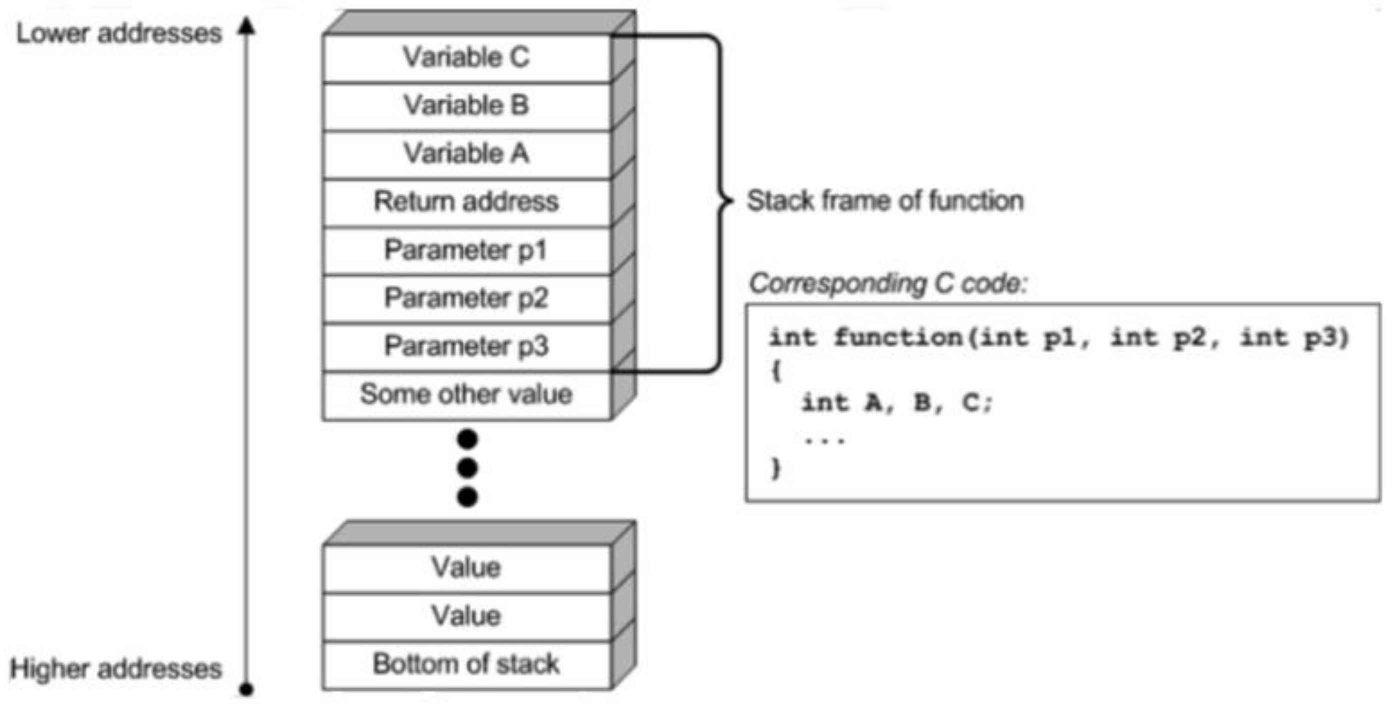

Higher addresses
bottom of stack
STACK segment (stack frames consisting of parameters, return addresses and local variables)
variable size
top of stack
free space
HEAP segment (dynamic variables, managed by malloc(), free() etc.)
variable size
BSS segment (uninitialized global and static variables)
fixed size
DATA segment (initialized global and static variables)
fixed size
TEXT segment (program code)
fixed size
Lower addresses
Virtual address space of a process

Stack

- A stack is a region of memory that is dynamically allocated to the program in a last-in, first-out (LIFO) pattern.
- A stack pointer (typically a register) contains the memory address of the top of the stack.
- When an item is pushed onto the stack, the stack pointer is decreased and the item is stored at the new location referenced by the stack pointer.
- When an item is popped off the stack, the item referenced by the stack pointer is (typically) copied into a register and the stack pointer is increased.

Stack

- It stores types of variables that have a fixed lifetime - local variables The stack is relatively small. It is generally not a good idea to do anything that eats up lots of stack space.



```
int function(int p1, int p2, int p3)
{
  int A, B, C;
  ...
}
```

Heap

- The heap segment keeps track of memory used for dynamic memory allocation.
- The heap starts from lower memory, growing up into higher memory. In C, when you use the new operator to allocate memory, this memory is allocated in the heap segment.
- Allocated memory stays allocated until it is specifically deallocated or the application ends (at which point the OS should clean it up).
- Because the heap is a big pool of memory, large arrays, structures, or classes can be allocated here

```
int *ptr = new int; // ptr is assigned 4 bytes in the heap
int *array = new int[10]; // array is assigned 40 bytes in ↩
    the heap
```

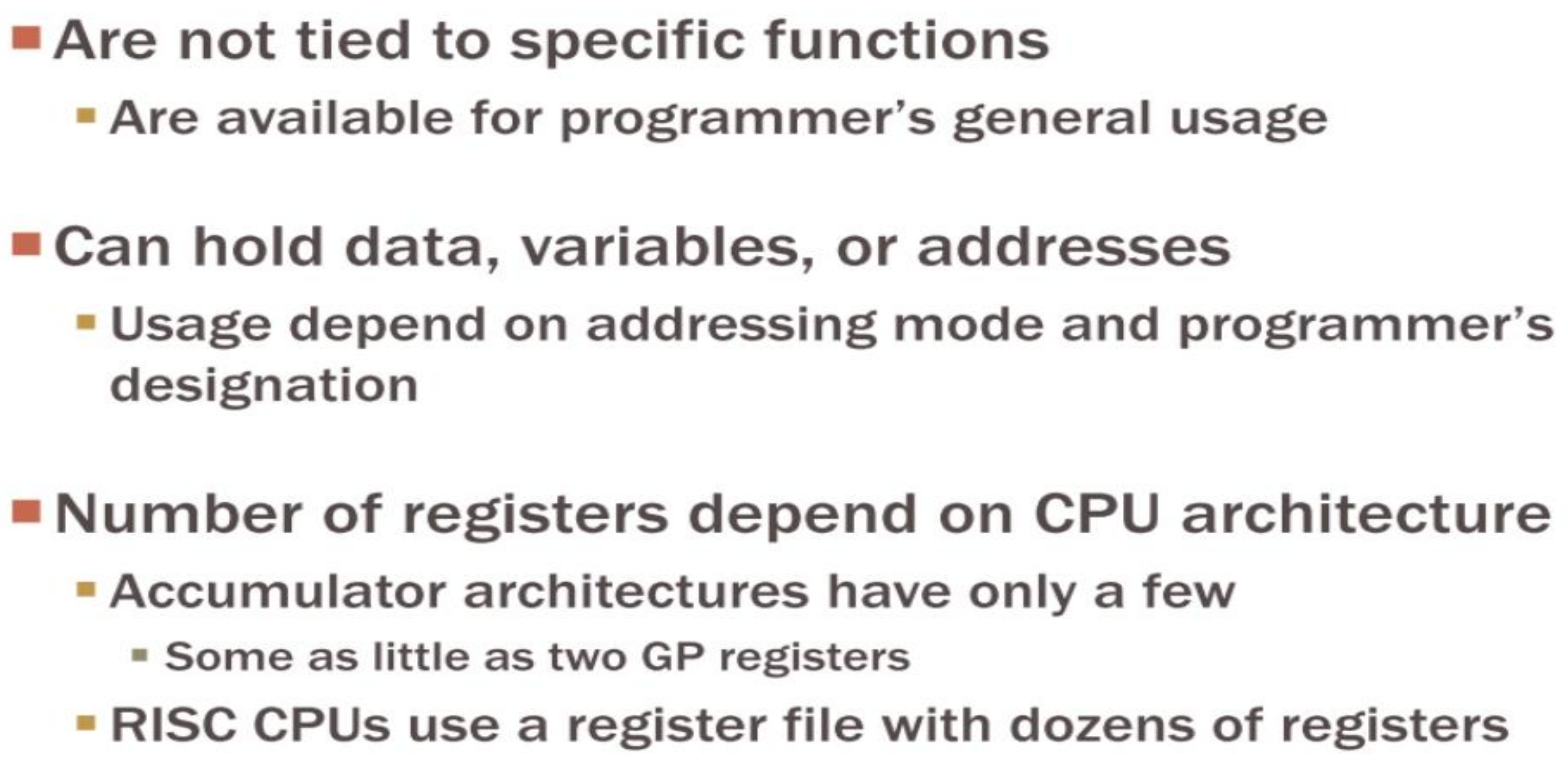
Memory Leak

- A garbage collector is a task that runs either periodically or when memory gets tight.
- It automatically frees any portions of memory that are no longer referenced.
- With or without garbage collection, it is possible for a program to inadvertently accumulate memory that is never freed.
- This is known as a memory leak.
- The program will eventually fail when physical memory is exhausted

Registers

- Provide temporary storage for:
  - Data & operands
  - Memory addresses
  - Control words
- Fastest form of storage
- Smallest Capacity
- Volatile Contents
  - Contents lost when CPU is de-energized
- Register Types
  - General Purpose
  - Special Purpose

General Purpose Registers



- Are not tied to specific functions
  - Are available for programmer's general usage
- Can hold data, variables, or addresses
  - Usage depend on addressing mode and programmer's designation
- Number of registers depend on CPU architecture
  - Accumulator architectures have only a few
    - Some as little as two GP registers
  - RISC CPUs use a register file with dozens of registers

Special Purpose Registers

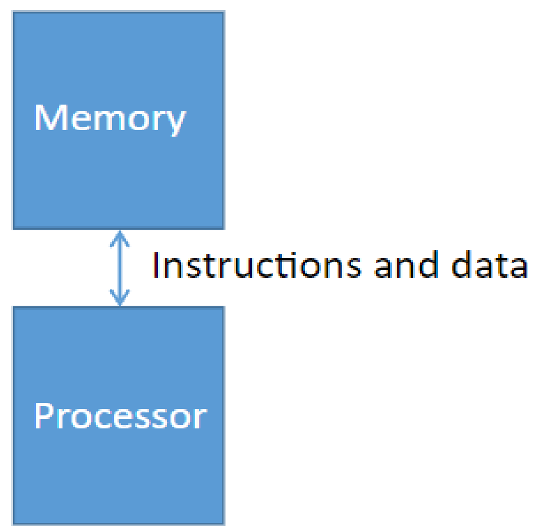
- Instruction Register (IR)

  Holds the instruction being currently decoded and executed
- Program Counter (PC)
  - Holds the address of the next instruction to be fetched from memory
- Stack Pointer (SP)
  - Holds the address of the current top-of-stack (TOS)
- Status Register (SR)
  - Holds the current CPU status
  - Status is indicated by a set of *flags*
  - A Flag: an individual bit indicating some condition

Memory Interface

Processor - Memory Interface

Memory must be **random access memory** - *individual* memory locations can be accessed in any order at the same high speed.



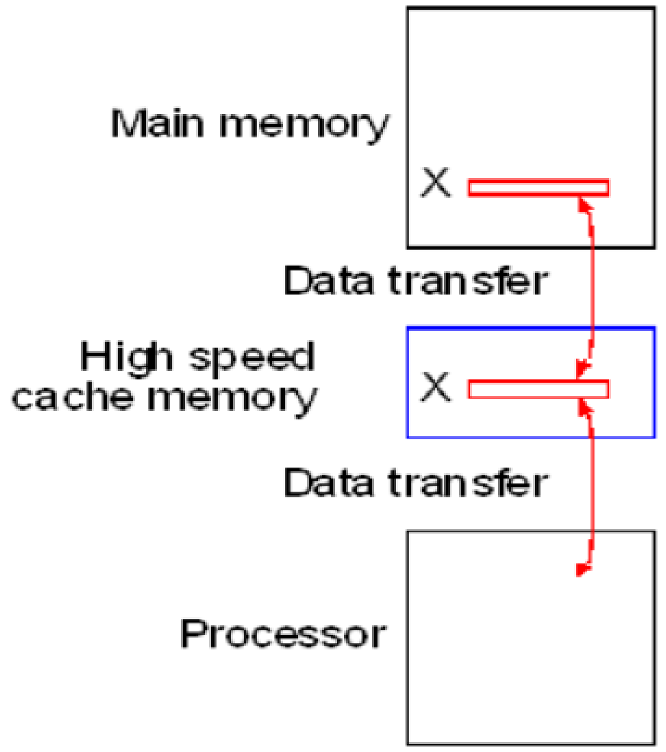
The memory that connects to the processor should operate preferably at a speed that matches the processor, so as not to slow the system down.

Large dynamic semiconductor RAM used for main memory cannot operate at that speed (much slower).

Relatively small static semiconductor memory can be designed to operate faster.

Cache Memory

A high speed memory called a cache memory placed between the processor and main memory, operating a speed closer to that of the processor.



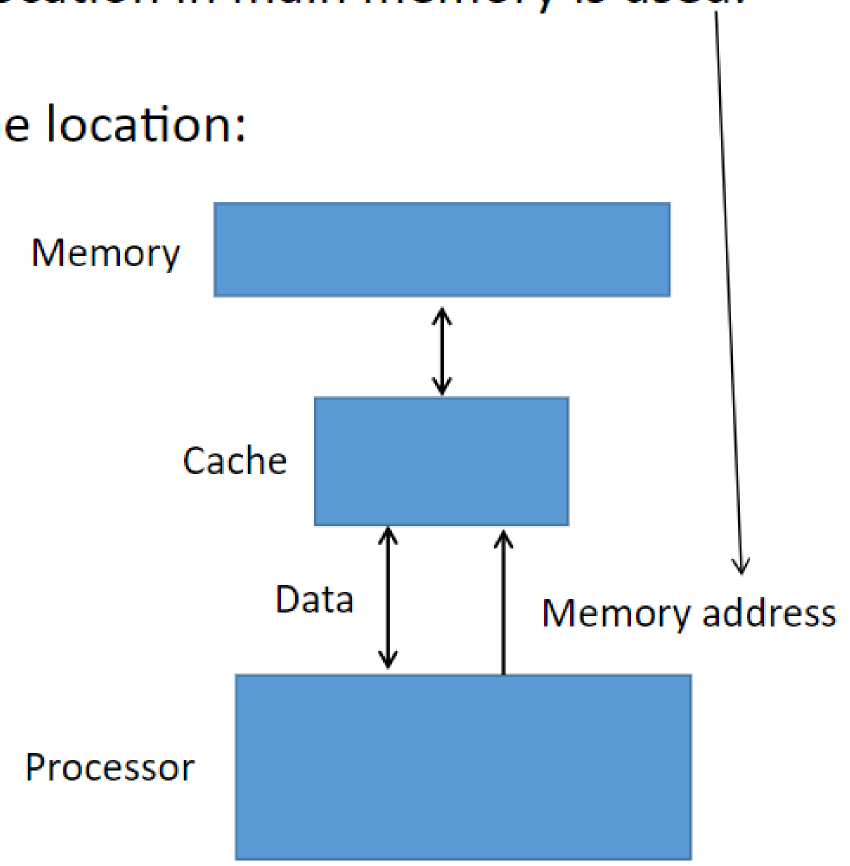
Information must be in cache memory for processor to access it:

Cache Memory Organizations

Need a way to select the location within the cache.
The memory address of its location in main memory is used.

Three ways of selecting cache location:

1. Fully associative
2. Direct mapped
3. Set associative



Thank You