Microprocessor System Design

EHB432E

Lecture -4

Prof. Dr. Müştak E. Yalçın

Istanbul Technical University

mustak.yalcin@itu.edu.tr

General-Purpose Processors: Software

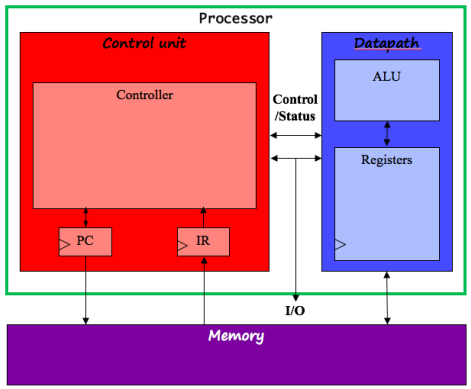
- Processor designed for a variety of computation tasks
- Low unit cost, in part because manufacturer spreads NRE over large numbers of units
  - Motorola sold half a billion 68HC05 microcontrollers in 1996 alone
- Carefully designed since higher NRE is acceptable
  - Can yield good performance, size and power
- Low NRE cost, short time-to-market/prototype, high flexibility
  - User just writes software; no processor design
- a.k.a. "microprocessor" – "micro" used when they were implemented on one or a few chips rather than entire rooms

BLG 212E Microprocessor Systems

General-Purpose Processors: Basic Architecture

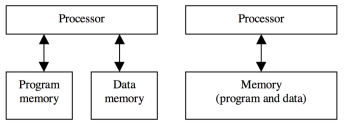
Key differences

- Datapath is general
- Control unit doesn't store the algorithm – the algorithm is "programmed" into the memory



General-Purpose Processors: Basic Architecture

- Datapath Unit: consists of circuitry for transforming data and for storing temporary data.
  N-bit processor : N-bit ALU, registers, buses, memory data interface
- Control Unit: consists of circuitry for retrieving program instructions and for moving data to, from and through the datapath according to those instr.
  Program Counter's size determines address space
- Memory: While registers serve a processor's short term storage requirements, memory serves the processor's medium and long-term information-storage requirements. Two memory architectures: Harvard and Princeton.



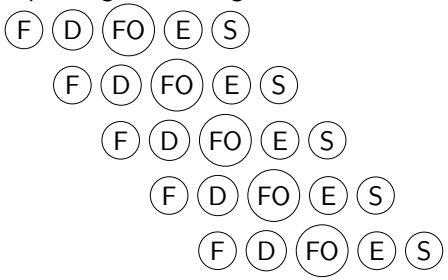
General-Purpose Processors: Operations

- Datapath Operations
  - Load : Read memory location into register
  - ALU operation: Input certain registers through ALU, store back in register
  - Store : Write register to memory location
- Control Unit
  - Control unit: configures the datapath operations
    - Sequence of desired operations ("instructions") stored in memory – "program"
  - Instruction cycle – broken into several sub-operations, each one clock cycle, e.g.:
    - Fetch: Get next instruction into IR
    - Decode: Determine what the instruction means
    - Fetch operands: Move data from memory to datapath register
    - Execute: Move data through the ALU
    - Store results: Write data from register to memory

Instruction Cycles

```
0:  INPUT S0,(S1);
          ;Fetch :  get next instruction into IR
          ;Decode:  Determine what the instruction means
          ;Fetch operands:  Move data from memory to datapath register
          ;Execute:  Move data through the ALU
          ;Store results:  Write data from register to memory
1:  ADD S0, 01;
          Fetch :  get next instruction into IR
          Decode:  Determine what the instruction means
          Fetch operands:  Move data from memory to datapath register
          Execute :Move data through the ALU
          Store results:  Write data from register to memory
2:  STORE S0,(S1)
          Fetch :  get next instruction into IR
          Decode:  Determine what the instruction means
          Fetch operands:  Move data from memory to datapath register
          Execute :Move data through the ALU
          Store results:  Write data from register to memory
```

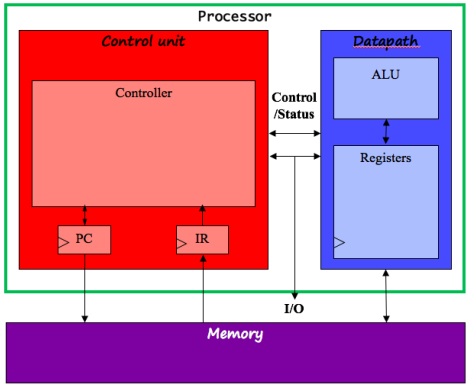

Pipelining: Increasing Instruction Throughput



General-Purpose Processors

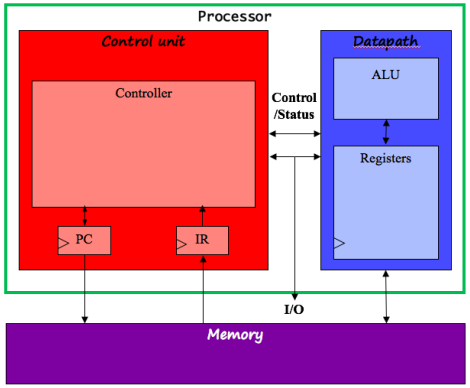
Performance can be improved by:

- Faster clock (but there is a limit)
- Pipelining: slice up instruction into stages, overlap stages
- Multiple ALUs to support more than one instruction stream



Architectural Considerations

- Faster clock (but there's a limit)
- Pipelining: slice up instruction into stages, overlap stages
- Multiple ALUs to support more than one instruction stream



Programmer's View

A programmer writes the program instructions that carry out the desired functionality on the general-purpose processor.

- Programmer doesn't need detailed understanding of architecture
  Instead, needs to know what instructions can be executed
- Two levels of instructions:
  - Assembly level
  - Structured languages (C, C++, Java, etc.)
- Most development today done using structured languages
  - But, some assembly level programming may still be necessary
  - Drivers: portion of program that communicates with and/or controls (drives) another device
    - Often have detailed timing considerations, extensive bit manipulation
    - Assembly level may be best for these

Programmer's View: Instruction set

The assembly-language programmer must know the processors's instruction set.
An instruction : an opcode field and operand fields.

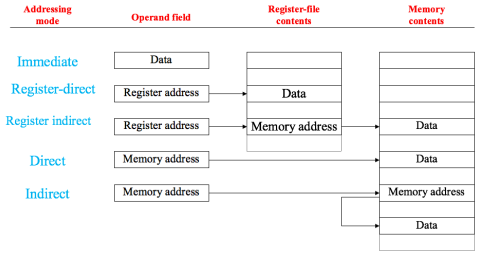
```
[opcode (specifies the operation)] [operand1 (loc.  of data)] [operand2 (loc.  of data)]
```

- Data-transfer instructions
- Arithmetic/logical instructions
- Branch instructions
  - Unconditional Jump
  - Conditional Jump
  - Return instruction
  - Call instruction (saves the add. of current inst.)

Source operands serve as input to the operation, while a destination operand stores the output.

Programmer's View: Instruction set

The operand field may indicate the data's location through one of several addressing modes :

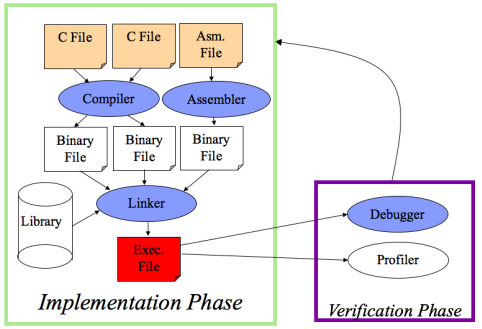


The embedded systems programmer must be aware of

- Program and data memory space
- Registers (for data-transfer instruction etc.): How many are there?
- Input and output (I/O) facilities (to help communicate with other devices)
- Interrupts :
  An interrupt causes the processor to suspend execution of the main program, and instead jump to an Interrupt Service Routine (ISR) that fulfills a special, short-term processing need.

Development Environment

- Development processor: on which we write and debug our programs
- Target processor : that the program will run on in our embedded system

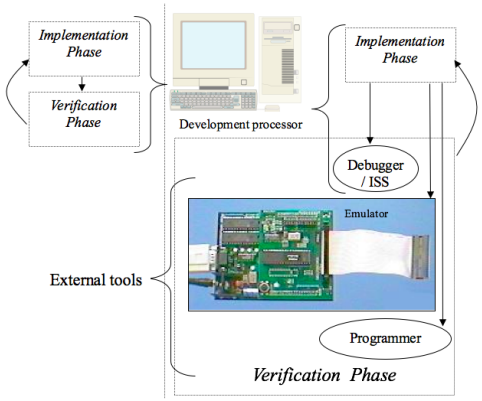


Development Environment

- Assemblers translate assembly instructions to binary machine instructions.
- A linker allows a programmer to create a program in separately-assembled files.
- Compilers translate structured programs into machine (or assembly) programs.
- A cross-compiler executes on one processor (our development processor), but generates code for a different processor (our target processor).
- Debuggers help programmers evaluate and correct their programs.
- Emulators support debugging of the program while it executes on the target processor.

All this tool in integrated development environment (IDE)

Development Environment



- Instruction set simulator (ISS):
  - Gives us control over time – set breakpoints, look at register values, set values, step-by-step execution, ...
  - But, doesn't interact with real environment