Jan 25, 2010

At 2005, 98% of all computers are embedded

- Programmable (8bit)
  - Microcontroller (2$)
  - General Purpose Computer(100$)
  - DSP (4$) ex. MAC, Multiple Accumulate $Y_N=C.X_N+Y_{N-1}$
- FPGA (10$)
  - Field Programmable Gate Array
  - Faster, reconfigurable, have lots of look up table
  - Uses Verilog HDL
- Custom Gates ($)
  - ASIC - most power efficient, one time programming, application specific
  - PLC

DSP vs. CPU

- Application specific instruction
- Real time
- Parallel processing
- SIMD – Single Instruction Multiple Data

Making a vending machine

1.) I/O
2.) Controller
3.) Sensor/ Actuator/ Interface
4.) Display
5.) Environment/ Specs
6.) Power

What to do to the microcontroller?

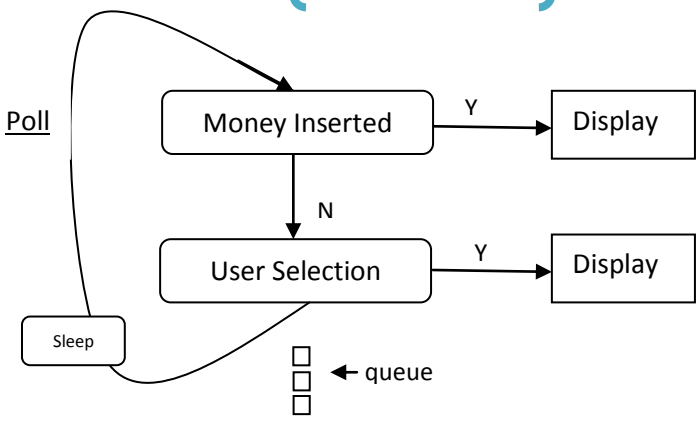
Input

```
int     Money Inserted ()
int     Price List (int i)
int     User Selection ()        // 0 means nothing
bool    Item in Stock (int i)
```

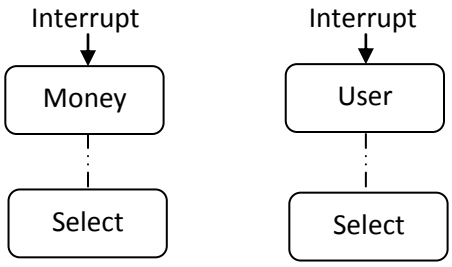
Output

```
void    Vend Item (int i)
```

Detecting Changes
- Interrupt
- Poll



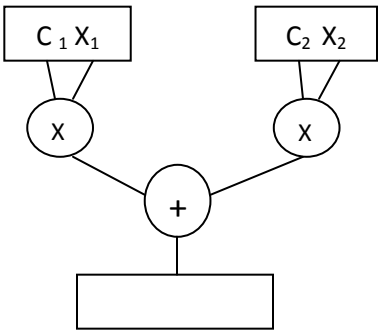
Interrupt



$$Y_n = \sum_{i=0}^{N-1} C_i \cdot X_{(n-i)}$$

$Y_n = C_1 \cdot X_1 + C_2 \cdot X_2 + C_3 \cdot X_3 \ldots . C_N \cdot X_N$

$Y_n = C_1 \cdot X_N + C_2 \cdot X_{(N-i)}$ ---- 4 reads, 1 write + instruction reads



4

36

1

4

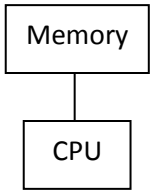
	Cycles
Mul	36
Add	1
Mem	4

Cycles needed

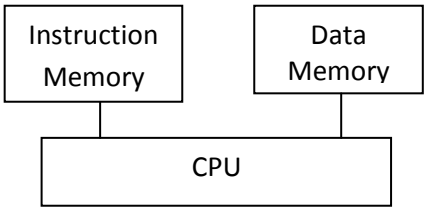
36 x 2
+ 1
+ 4x4
+ 4

Architectures

Von Neumann (Princeton)



Harvard



Feb. 9, 2010

Models of Embedded Systems

1.) Flow Chart
2.) Finite State Machine

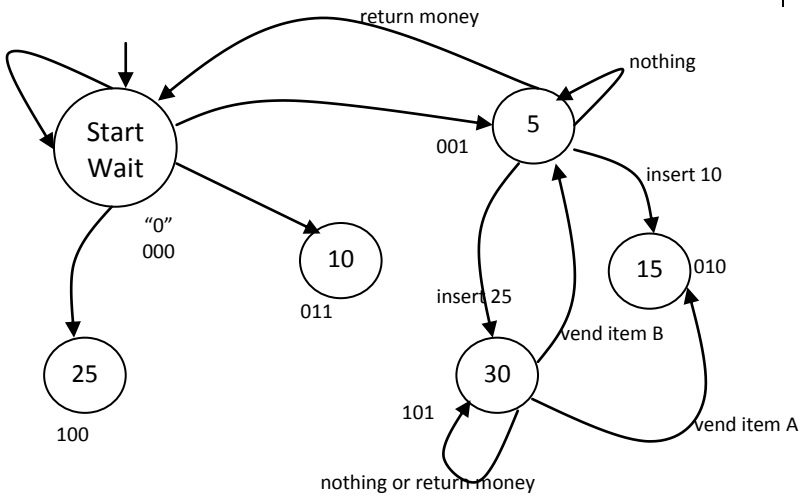
Ex. Vending Machine

Item A- 15 ¢ D - F- Return

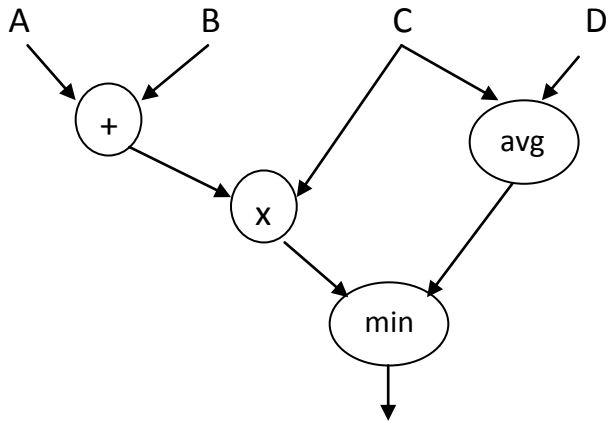
Item B -25 ¢ E -

Accepts 5, 10, 25 ¢ coins

Inputs	Current	Next	Outputs
ABCDEF			
000000	000	000	
100000	000	001	



3.) Control Data Flow (CDFG)



Useful for continuous repetitive process

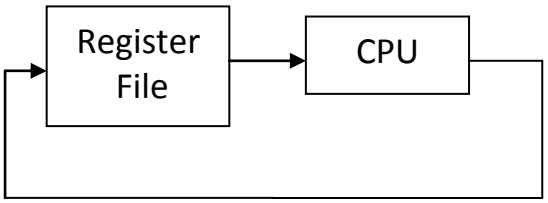
Attributes of Embedded Systems

1.) Synchronous / Asynchronous - in terms of inputs and internals, sequential vs. concurrent processing (parallel, simultaneous)
2.) Constraints on outputs- real time
3.) Finite State/ Infinite State
4.) Continuous, Sampled, Discrete digital
5.) Communication
6.) Composability- tools, compilers, debuggers
7.) Deterministic vs. Non-deterministic

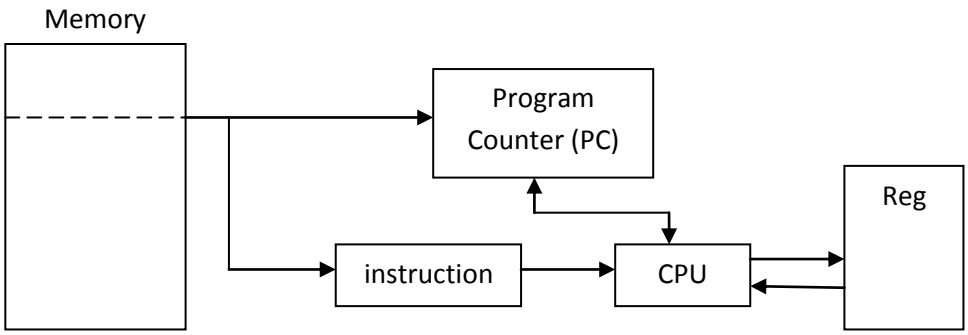
Validate

- Testing input/output
- Simulation
- By construction

RISC vs CISC (Reduced Instruction Set vs. Complex ISC)

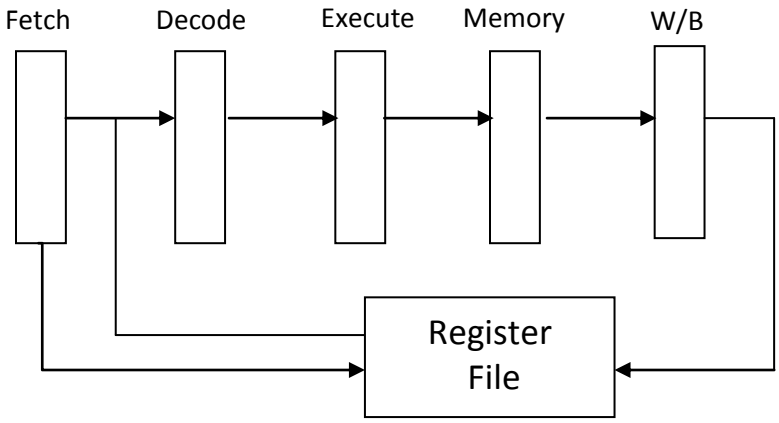


ADD	R_{S1}, R_{S2}, R_{SD}	// $R_{SD} = R_{S1} + R_{S2}$
SUB		
LOAD	R, M_{ADDR}	// R=Memory [ADDR]
STORE	R, M_{ADDR}	
J	Address	// Jump address
JLE	R, Address	// if R<0, J address
JI	R	// Jumps to address in R



5 Stage Pipeline

1.) Fetch
2.) Decode
3.) Execute
4.) Memory
5.) Write Back



① LOAD R_1 Mem [0000]

② ADD R_1, R_2, R_3

③ SUB R_5, R_6, R_8

Interchanging will save cycles

Sources Destination

In RISC, 1 instruction/cycle

Wrong version

Time	F	D	E	M	W
0	①				
1	②	①			
2	③	x ②	①		
3		③	②	①	
4			③	②	①
5				③	②
6					③

Wrong value

Correct version

Time	F	D	E	M	W
0	①				
1	②	①			
2	②	②	①		
3	③	②		①	
4	③	②			①
5		③	②		
6			③	②	
7				③	②
8					③

Done in 9cycles