

## Arquitetura de Sistemas Embarcados

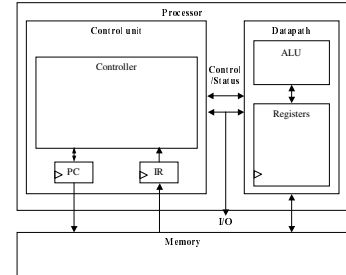
Edna Barros (ensb@cin.ufpe.br)



Centro de Informática – UFPE

## Arquitetura Básica

- Unidade de Controle e de Processamento
- Diferenças
  - Unidade de Processamento é genérica
  - Unidade de controle não armazena algoritmo (memória)

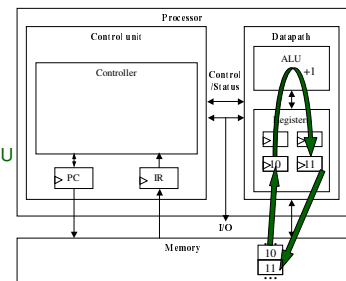


Arquitetura de Sistemas Embarcados 4

## Capítulo 3 Processadores de Propósito Geral: Software

## Operações de Processamento

- Load
  - Cópia de memória em registrador
- Operação na ALU
  - Valores em Registradores são processados pela ALU e armazenados em registrador
- Store
  - Cópia de registrador em memória



Arquitetura de Sistemas Embarcados 5

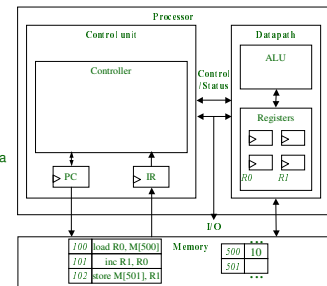
## Introdução

- Processadores de Propósito Geral
  - Processador projetado para uma variedade de tarefas computacionais
  - Baixo custo unitário (Custo NRE para grande volume)
    - Motorola vendeu meio bilhão de microcontroladores 68HC05 em 1996
  - Cuidadosamente projetado
    - Otimiza desempenho, tamanho e potência
  - Reduzido time-to-market (protótipo), grande flexibilidade
    - Usuário apenas descreve o software
  - “microprocessador” – “micro” usado devido a ser implementado com poucos circuitos

Arquitetura de Sistemas Embarcados 3

## Unidade de Controle

- Unidade de Controle: configura operações do datapath
  - Sequência de operações (instruções) desejadas armazenadas na memória (programa)
- Ciclo de Instrução – várias sub-operações (cada uma em um ciclo de relógio)
  - Busca: armazena instrução em IR, atualiza PC
  - Decodificação: determina o que a instrução significa
  - Busca de Operandos: cópia de dados da memória para registradores na unidade de Processamento
  - Execução: Processa dados na ALU
  - Armazena resultados: escreve resultados de registrador na memória



Arquitetura de Sistemas Embarcados 6

### Sub-operações da Unidade de Controle

- Busca
  - Cópia da instrução em IR
  - PC: Contador de programa aponta para próxima instrução
  - IR: armazena instrução que foi buscada

Arquitetura de Sistemas Embarcados 7

### Sub-operações da Unidade de Controle

- Execução
  - Processa dados na ALU
  - (Para esta instrução em particular nada acontece durante esta sub-operação)

Arquitetura de Sistemas Embarcados 10

### Sub-operações da Unidade de Controle

- Decodificação
  - Determina significado da instrução

Arquitetura de Sistemas Embarcados 8

### Sub-operações da Unidade de Controle

- Armazena resultados
  - Escreve dado de registrador em memória
  - (Para esta instrução em particular nada acontece durante esta sub-operação)

Arquitetura de Sistemas Embarcados 11

### Sub-operações da Unidade de Controle

- Busca de Operandos
  - Cópia de dados da memória para registradores no datapath

Arquitetura de Sistemas Embarcados 9

### Ciclos de uma Instrução

PC=100  
Fetch Decode Fetch Exec. Store results

Arquitetura de Sistemas Embarcados 12

### Ciclos de uma Instrução

Arquitetura de Sistemas Embarcados 13

### Considerações da Arquitetura

- **Frequência do Clock**
  - Deve ser maior que o maior retardo de carregamento de registrador
  - Acesso à memória possui o maior retardo

Arquitetura de Sistemas Embarcados 16

### Ciclos de uma Instrução

Arquitetura de Sistemas Embarcados 14

### Pipelining: Aumentando Taxa de Execução das Instruções

Arquitetura de Sistemas Embarcados 17

### Considerações da Arquitetura

- **Processador de N-bits**
  - ALU, registradores, barramento, interface de memória N-bits
  - Comum em aplic. emb: 8-bit, 16-bit, 32-bit
  - Comum em Desktop/servidores: 32-bit, ou 64
- **Tamanho do PC determina espaço de endereçamento**

Arquitetura de Sistemas Embarcados 15

### Multi-ciclo vs. Pipeline

Arquitetura de Sistemas Embarcados 18

## Arquiteturas Superescalares e VLIW

- Desempenho pode ser melhorado por:
  - Clock mais rápido
  - Pipelining
  - Múltiplas ALUs permitindo a execução de uma de uma sequência de instruções
    - Superescalares
      - Escalar: operações não vetoriais
      - Instruções buscadas em blocos e se executa o maior número possível
        - Hardware adicional para descobrir instruções independentes
    - VLIW
      - cada palavra na memória tem múltiplas instruções independentes
        - Compilador deve descobrir e escalar instruções
        - Aumento de popularidade

Arquitetura de Sistemas Embarcados 19

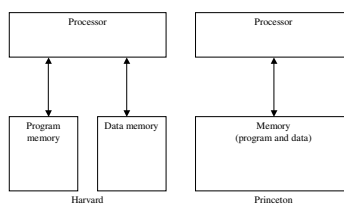
## Sistema Hierárquico de Memória

Speed	CPU	Size	Cost (\$/bit)
Fastest	Memory	Smallest	Highest
Slowest	Memory	Biggest	Lowest

Arquitetura de Sistemas Embarcados 22

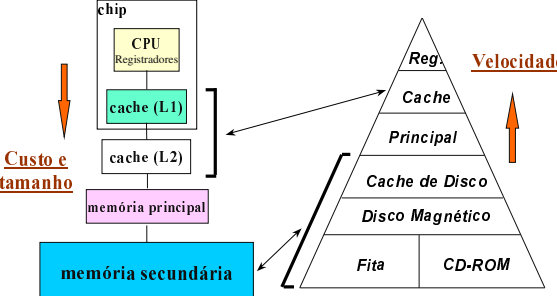
## Arquiteturas de Memória

- Princeton
  - Menos fios
- Harvard
  - Acesso simultâneo à dado e instruções



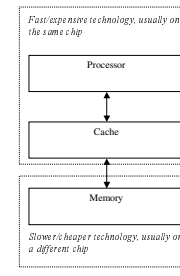
Arquitetura de Sistemas Embarcados 20

## Hierarquia de Memória



## Memória Cache

- Acesso à memória pode ser lento
- Cache é pequena porém tão rápida quanto o processador
  - Possui cópia de parte da memória
  - Hits e misses



Arquitetura de Sistemas Embarcados 21

## Visão do Programador

- Programador não necessita visão detalhada da implementação
  - Precisa saber quais instruções podem ser executadas
- Dois Níveis de Abstração
  - Linguagem de Montagem (Assembly level)
  - Linguagens Estruturadas (C, C++, Java, etc.)
- Projetos são feitos usando linguagens estruturadas
  - MAS algum nível de assembly PODE ser necessário
  - Drivers: parte do programa que se comunica com e/ou controla outro dispositivo
    - Com considerações detalhadas de tempo e manipulação de bits
    - Nível de assembly pode ser o ideal

Arquitetura de Sistemas Embarcados 24

## Instruções Nível Assembly

Instruction 1	opcode	operand1	operand2
Instruction 2	opcode	operand1	operand2
Instruction 3	opcode	operand1	operand2
Instruction 4	opcode	operand1	operand2
	...		

- Repertório de Instruções:
  - Define o conjunto de instruções que o processador pode executar
    - Transferência de Dado: memória/registorador, registorador/registorador, I/O, etc.
    - Aritméticas e Lógicas
    - Desvios: determina outro valor para PC (diferente de PC+1)

Arquitetura de Sistemas Embarcados 25

## Exemplos de Programas

<pre>C program int total = 0; for (int i=10; i!=0; i--)     total += i; // next instructions...</pre>	<pre>Equivalent assembly program 0  MOV R0, #0; // total = 0 1  MOV R1, #10; // i = 10 2  MOV R2, #1; // constant 1 3  MOV R3, #0; // constant 0 Loop: JZ R1, Next; // Done if i=0 5  ADD R0, R1; // total += i 6  SUB R1, R2; // i-- 7  JZ R3, Loop; // Jump Next: always // next instructions...</pre>
---	--

- Tente outros...
  - Handshake: Espere até que o valor de M[254] não seja 0, faça M[255] igual 1, espere até que M[254] seja, faça M[255] igual a 0 (assuma que estes endereços são portas).
  - Conte as ocorrências de zeros em um array armazenado na memória (endereços de 100 a 199)

Arquitetura de Sistemas Embarcados 28

## Um Repertório de Instruções Simples

Assembly instruct.	First byte	Second byte	Operation
MOV Rn, direct	0000 Rn	direct	Rn = M(direct)
MOV direct, Rn	0001 Rn	direct	M(direct) = Rn
MOV @Rn, Rm	0010 Rn	Rm	M(Rn) = Rm
MOV Rn, #immed.	0011 Rn	immediate	Rn = immediate
ADD Rn, Rm	0100 Rn	Rm	Rn = Rn + Rm
SUB Rn, Rm	0101 Rn	Rm	Rn = Rn - Rm
JZ Rn, relative	0110 Rn	relative	PC = PC + relative (only if Rn is 0)

Arquitetura de Sistemas Embarcados 26

## Visão do Programador

- Espaço de Programa e de Dados
  - Processadores embarcados são bastantes limitados (em geral)
    - e.g., 64 Kbytes de programa, 256 bytes de RAM (expansível)
- Registradores: Quantos existem?
  - Somente para programadores em assembly
- I/O
  - Como se comunicar com sinais externos?
- Interrupções

Arquitetura de Sistemas Embarcados 29

## Modos de Endereçamento

Addressing mode	Operand field	Register-file contents	Memory contents
Immediate	Data		
Register-direct	Register address	Data	
Register indirect	Register address	Memory address	Data
Direct	Memory address		Data
Indirect	Memory address		Memory address Data

Arquitetura de Sistemas Embarcados 27

## Exemplo: driver de porta paralela

LPT Connection Pin	I/O Direction	Register Address
1	Output	0 <sup>th</sup> bit of register #2
2-9	Output	0 <sup>th</sup> bit of register #2
10,11,12,13,15	Input	6,7,5,4,3 <sup>rd</sup> bit of register #1
14,16,17	Output	1,2,3 <sup>rd</sup> bit of register #2

- Usando linguagem assembly pode-se configurar a porta paralela de um PC para se fazer E/S digital
  - Escrita e leitura de 3 registradores especiais (conforme tabela que mostra pinos do conector da porta e sua posição nos registradores)
  - Exemplo: porta paralela monitora uma chave e acende ou apaga um LED de acordo com a posição da chave

Arquitetura de Sistemas Embarcados 30

## Exemplo Porta Paralela

```

; This program consists of a sub-routine that reads
; the state of the input pin, determining the on/off state
; of our switch and asserts the output pin, turning the LED
; on/off accordingly
;---
CheckPort proc
    push    dx                ; save the content
    push    ax                ; save the content
    mov     dx, 3BCh          ; base + 1 for register #1
    in     al, dx             ; read register #1
    and    al, 10h           ; mask out all but bit # 4
    cmp    al, 0             ; is it 0?
    jne    SwitchOn          ; If not, we need to turn the LED on
;---
SwitchOff:
    mov     dx, 3BCh + 0;    ; base + 0 for register #0
    in     al, dx           ; read the current state of the port
    and    al, 07h          ; clear first bit (masking)
    out    dx, al           ; write it out to the port
    jmp    Done
;---
SwitchOn:
    mov     dx, 3BCh + 0;    ; base + 0 for register #0
    in     al, dx           ; read the current state of the port
    or     al, 01h          ; set first bit (masking)
    out    dx, al           ; write it out to the port
;---
Done:
    pop    dx                ; restore the content
    pop    ax                ; restore the content
CheckPort endp
    
```

LPT Connection Pin	I/O Direction	Register Address
1	Output	0 <sup>th</sup> bit of register #2
2-9	Output	0 <sup>th</sup> bit of register #2
10,11,12,15	Input	0, 7, 5, 4, 3 <sup>rd</sup> bit of register #2
14,16,17	Output	1, 2, 3 <sup>rd</sup> bit of register #2

Arquitetura de Sistemas Embarcados 31

## Processo de Desenvolvimento de Software

- **Compiladores**
  - Cross compiler
  - Executa em um processador mas gera código para outro
- **Assemblers**
- **Linkers**
- **Debuggers**
- **Profilers**

Arquitetura de Sistemas Embarcados 34

## Sistema Operacional

- Software opcional que fornece serviços de “baixo nível” para a aplicação
  - Gerenciamento de arquivos e acesso a disco
  - Interface entre teclado e vídeo
  - Escalonamento de múltiplos programas para execução
    - Ou mesmo múltiplas threads de um programa
  - Programa faz chamadas ao S.O.

```

DB file_name "out.txt" -- store file name
MOV RO, 1324 -- system call "open" id
MOV R1, file_name -- address of file-name
INT 34 -- cause a system call
JZ RO, L1 -- if zero -> error
; . . . read the file
JMP L2 -- bypass error cond.
L1:
; . . . handle the error
L2:
    
```

Arquitetura de Sistemas Embarcados 31

## Executando um programa

- Se processador de desenvolvimento é diferente do processador alvo, como executar o código compilado???
- Duas opções:
  - Download para processador alvo
  - Simulação
- Simulação
  - Um método: Descrever o processador usando HDL (Hardware description language)
    - Lento
  - Outro Método: Usar ISS (*Instruction set simulator*)
    - Executa no processador de desenvolvimento as instruções do processador alvo

Arquitetura de Sistemas Embarcados 34

## Ambiente de Desenvolvimento

- **Processador de Desenvolvimento**
  - Processador usado na escrita e depuração de programas
    - Usualmente PC
- **Processador Target**
  - Processador que irá executar o programa no sistema embarcado
    - Usualmente diferentes do processador de desenvolvimento

Arquitetura de Sistemas Embarcados 33

## Instruction Set Simulator para o Processador Simples

```

#include <stdio.h>
typedef struct {
    unsigned char first_byte, second_byte;
} instruction;

instruction program[1024]; //Instruction memory
unsigned char memory[256]; //data memory

void run_program(int num_bytes) {
    int pc = -1;
    unsigned char reg[16], fb, sb;
    while( ++pc < (num_bytes / 2) ) {
        fb = program[pc].first_byte;
        sb = program[pc].second_byte;
        switch( fb >> 4 ) {
            case 0: reg[fb & 0x0f] = memory[sb]; break;
            case 1: memory[reg[fb & 0x0f]] = reg[sb & 0x0f]; break;
            case 2: reg[fb & 0x0f] = reg[sb & 0x0f]; break;
            case 3: reg[fb & 0x0f] = sb; break;
            case 4: reg[fb & 0x0f] += reg[sb >> 4]; break;
            case 5: reg[fb & 0x0f] -= reg[sb >> 4]; break;
            case 6: pc += sb; break;
            default: return -1;
        }
    }
}

int main(int argc, char *argv[]) {
    FILE* f;
    if( argc != 2 || (f = fopen(argv[1], "rb")) == NULL )
        return -1;
    if( run_program(fread(program, sizeof(program) == 0) ? print_memory_contents(); return(0); else return(-1); }
}
    
```

Arquitetura de Sistemas Embarcados 38

## Teste e Depuração

- ISS
  - Permite controlar tempo de execução, setar break-points, ver e setar valores de registradores, execução passo-a-passo,...
  - POREM não permite interação com ambiente REAL
- Download para placa
  - Uso de programador de dispositivo
  - Executa em ambiente real porém com baixa controlabilidade
- Compromisso: emulador
  - Executa em ambiente real (ou próximo do real)
  - Suporta alguma controlabilidade (via PC)

Arquitetura de Sistemas Embarcados 37

## Exemplo de ASIP: Digital Signal Processors (DSP)

- Para aplicações de processamento de sinais
  - Grande quantidade de dado digitalizado (em streaming)
  - Transformações nos dados devem ser rápidas
  - Ex: filtro para telefone celular, TV digital, sintetizador de música
- Características dos DSPs
  - Várias unidades de execução
  - Instrução Multiplicação-acumula
  - Operações eficientes em vetores, ex: add dois arrays
    - ALUs vetoriais, loop buffers, etc...

Arquitetura de Sistemas Embarcados 40

## Processadores de Repertório de Instruções de Aplicação Específica (ASIPs)

- Processadores de propósito geral
  - Algumas vezes MUITO GENÉRICOS para ser eficiente para aplicação específica
    - Ex: processamento de vídeo: buffers e operações em arrays
  - MAS processador de aplicação única possui alto custo NRE e não é programável
- ASIPs – Aplicam-se a um domínio particular
  - Possuem características de arquitetura específicas ao domínio
    - Ex: controle embarcado, processamento digital de sinais, processamento de vídeo, processamento de rede, telecomunicações, etc.
  - Suportam uma área de aplicação
  - São Programáveis

Arquitetura de Sistemas Embarcados 38

## ASIPs customizados

Processadores adquiridos como módulo de propriedade intelectual (IP)

- Modelo VHDL sintetizável
- Layout
- Possibilidade de ajustar hardware (adicionar ou retirar instruções)
  - Grande impacto no desempenho, tamanho e potência
  - Problema: necessidade de compilador/depurador para ASIP customizado
    - Uma solução: geração automática de compiladores/depuradores
      - e.g., [www.tensillica.com](http://www.tensillica.com)
    - Outra solução: retargettable compilers
      - e.g., [www.improvsys.com](http://www.improvsys.com) (customized VLIW architectures)

Arquitetura de Sistemas Embarcados 41

## Exemplo de ASIP :Microcontrolador

- Para aplicações de controle
  - Leitura de sensores, ativar atuadores
  - Manipula com eventos: existe dados mas em pequena quantidade
  - Ex: disk drive, máquina de lavar, forno de microondas, etc...
- Características dos Microcontroladores
  - Periféricos em Único CHIP
    - Temporizadores, conversores analógico-digital, comunicação serial, etc...
    - Acesso através de registradores (facilidade de programação)
  - Programa de memória e dados no chip
  - Acesso a vários dos pinos do chip
  - Instruções para manipulação de bits e operações de baixo-nível

Arquitetura de Sistemas Embarcados 39

## Selecionando o microprocessador

- Características
  - Técnicas: velocidade, potência, tamanho, custo
  - Outras: ambiente de desenvolvimento, conhecimento prévio, licenças, ...
- Como avaliar o desempenho de um processador?
  - Velocidade do Clock: nr. de instruções por ciclo pode diferir
  - Instruções por segundo: tipo de instruções podem diferir
  - Dhrystone Benchmark Sintético, desenvolvido em 1984. Dhrystones/sec.
    - MIPS: 1 MIPS = 1757 Dhrystones per second (based on Digital's VAX 11/780).
    - 750 MIPS = 750\*1757 = 1,317,750 Dhrystones per second
  - SPEC: conjunto de benchmarks mais realísticos orientados a desktop
  - EEMBC – EDN Embedded Benchmark Consortium, [www.eembc.org](http://www.eembc.org)
    - Benchmarks: automóveis, eletrônica de consumo, redes, automação de escritório, telecomunicações.

Arquitetura de Sistemas Embarcados 42

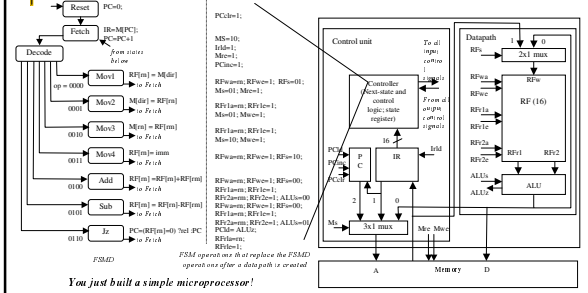
## Alguns Processadores

Processor	Clock speed	Periph.	Bus Width	MIPS	Power	Trans.	Price
General Purpose Processors							
Intel PIII	1GHz	2x16 K L1, 256K L2, MMX	32	~900	97W	-7M	\$900
IBM PowerPC 750X	550 MHz	2x32 K L1, 256K L2	32/64	~1300	3W	-7M	\$900
MIPS R5000	250 MHz	2x32 K L1, 256K L2, 2 way set assoc.	32/64	NA	NA	3.6M	NA
StrongARM SA-110	233 MHz	None	32	268	1W	2.1M	NA
Microcontroller							
Intel 8051	12 MHz	4K ROM, 128 RAM, 32 I/O, Timer, UART	8	-1	-0.2W	-10K	\$7
Motorola 68HC811	3 MHz	4K ROM, 192 RAM, 32 I/O, Timer, WDT, SPI	8	-5	-0.1W	-10K	\$5
Digital Signal Processors							
TI C5416	160 MHz	128K SRAM, 3 T1 Ports, DMA, 13 ADC, 9 DAC	16/32	-600	NA	NA	\$34
Lucent DSP32C	80 MHz	16K Inst., 2K Data, Serial Ports, DMA	32	40	NA	NA	\$75

Sources: Intel, Motorola, MIPS, ARM, TI, and IBM Web site/Datasheet; Embedded Systems Programming, Nov. 1998

Arquitetura de Sistemas Embarcados 43

## Um Microprocessador Simples



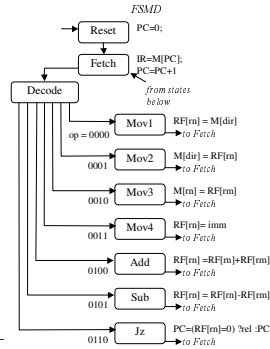
Arquitetura de Sistemas Embarcados 46

## Projetando um Processador de Propósito Geral

- Interessante para entender funcionamento
- Projeto é feito de forma bottom-up e mais otimizado

Declarations:  
bit PC[16], IR[16];  
bit M[64][16], RF[16][16];

Aliases:  
op IR[15..12] dir IR[7..0]  
rn IR[11..8] imm IR[7..0]  
rm IR[7..4]



Arquitetura de Sistemas Embarcados 47

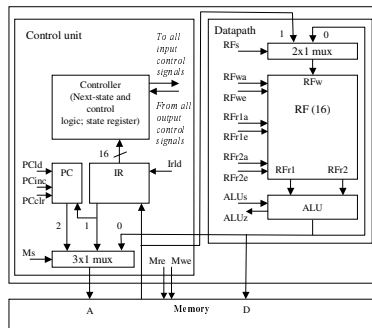
## Resumo

- Processadores de Propósito Geral
  - Bom desempenho, baixo custo NRE, flexível
- Controle, processamento e memória
- Uso de linguagens estruturadas
- Várias ferramentas disponíveis
  - Simulador ISS e emuladores
- ASIPs
  - Microcontroladores, DSPs, processadores de rede, ASIPs customizados
- Escolha do processador é uma etapa importante
- Projetar um processador de propósito geral é conceitualmente o mesmo que projetar um processador de propósito específico

Arquitetura de Sistemas Embarcados 47

## Um Microprocessador Simples

- Registradores para cada variável
- Unidades Funcionais para cada operação aritmética
  - Uso de uma ALU
- Conexões entre componentes
- Sinais de controle



Arquitetura de Sistemas Embarcados 45