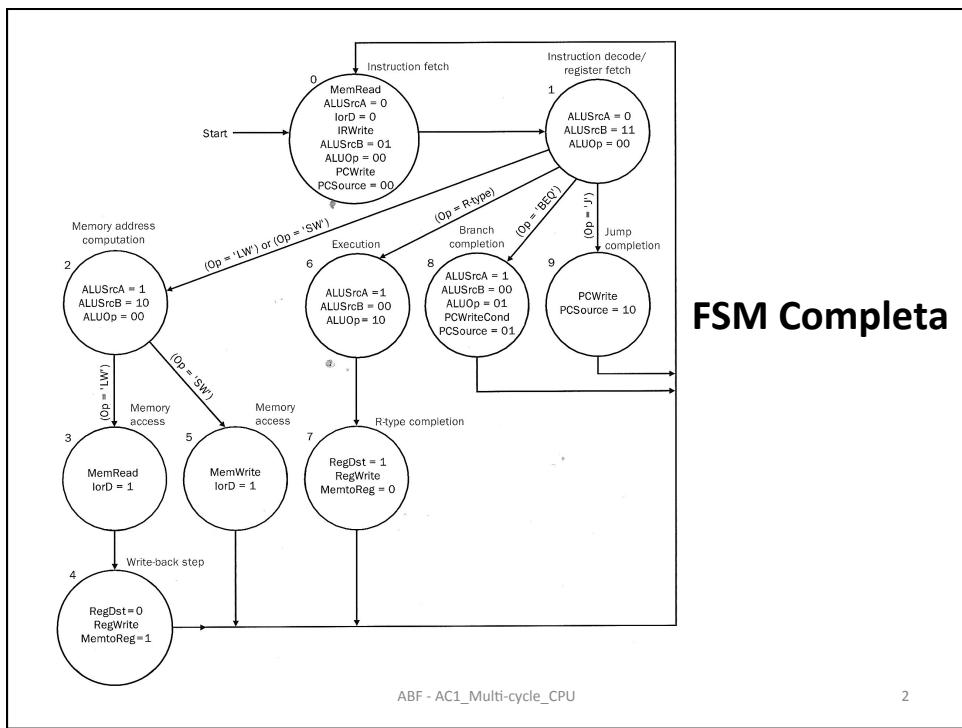
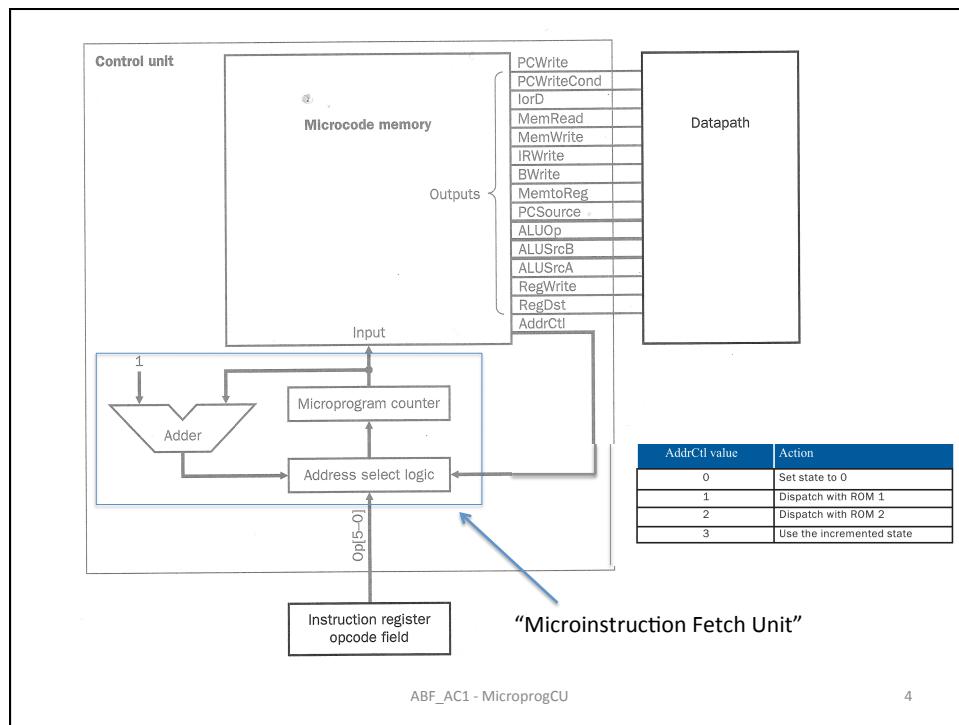
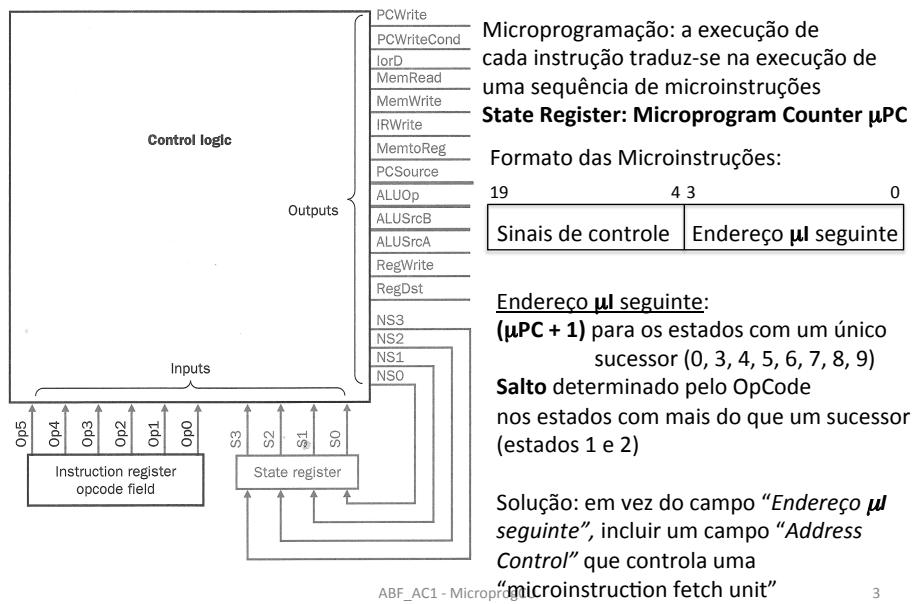


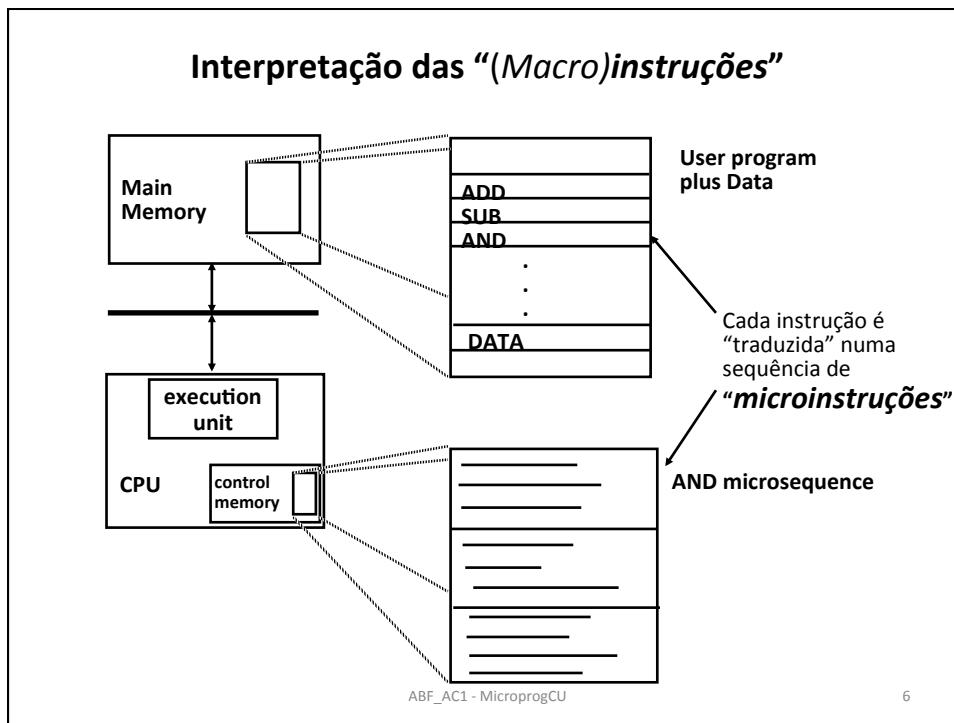
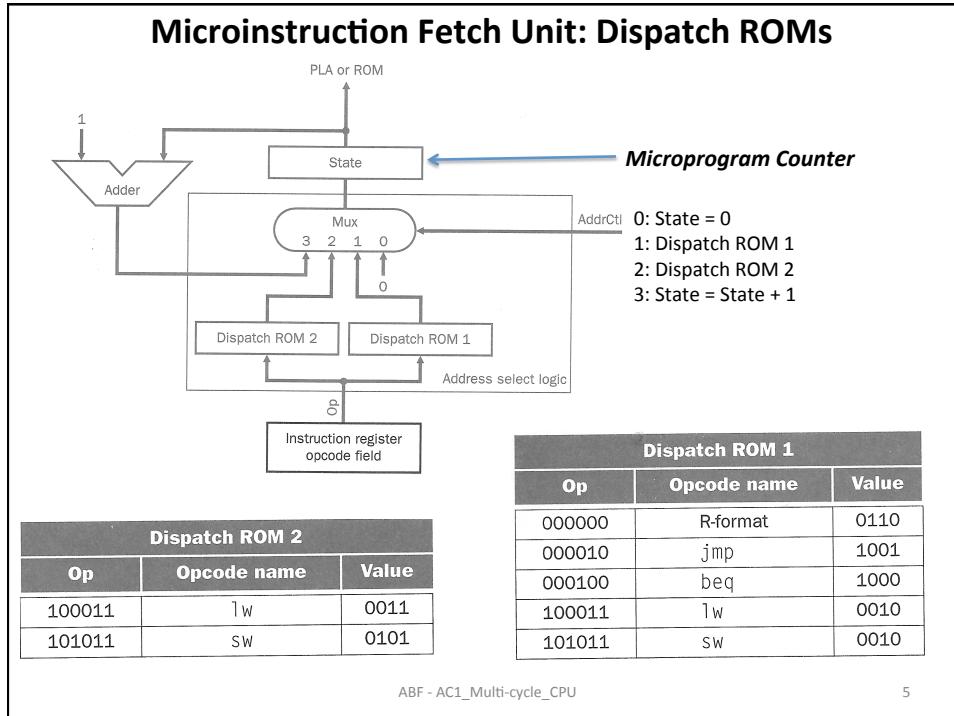
Processador Microprogramação

António de Brito Ferrari
ferrari@ua.pt



Unidade de Controle





AddrCtl

State number	Address-control action	Value of AddrCtl
0	Use incremented state	3
1	Use dispatch ROM 1	1
2	Use dispatch ROM 2	2
3	Use incremented state	3
4	Replace state number by 0	0
5	Replace state number by 0	0
6	Use incremented state	3
7	Replace state number by 0	0
8	Replace state number by 0	0
9	Replace state number by 0	0

Conceção do Microinstruction Set

- 1) Ponto de partida: lista dos sinais de control
- 2) Agrupar os sinais, de um modo lógico em “**microinstruction fields**”
- 3) Colocar os campos da microinstrução numa ordem lógica (e.g., ALU operation & ALU operands primeiro e microinstruction sequencing por último)
- 4) Criar uma representação simbólica para o formato das microinstruções, dando nome aos valores dos campos e que valores dão aos sinais de controle
- 5) Minimizar o comprimento da microinstrução, codificando as operações que nunca ocorrem simultaneamente

Formato das Microinstruções

Field name	Function of field
ALU control	Specify the operation being done by the ALU during this clock; the result is always written in ALUOut.
SRC1	Specify the source for the first ALU operand.
SRC2	Specify the source for the second ALU operand.
Register control	Specify read or write for the register file, and the source of the value for a write.
Memory	Specify read or write, and the source for the memory. For a read, specify the destination register.
PCWrite control	Specify the writing of the PC.
Sequencing	Specify how to choose the next microinstruction to be executed.

Field name	Values for field	Function of field with specific value
Label	Any string	Used to specify labels to control microcode sequencing. Labels that end in a 1 or 2 are used for dispatching with a jump table that is indexed based on the opcode. Other labels are used as direct targets in the microinstruction sequencing. Labels do not generate control signals directly but are used to define the contents of dispatch tables and generate control for the Sequencing field.
ALU control	Add Subt Func code	Cause the ALU to add. Cause the ALU to subtract; this implements the compare for branches. Use the instruction's funct field to determine ALU control.
SRC1	PC A B	Use the PC as the first ALU input. Register A is the first ALU input. Register B is the second ALU input.
SRC2	4 Extend Extshft Read	Use 4 for the second ALU input. Use output of the sign extension unit as the second ALU input. Use the output of the shift-by-two unit as the second ALU input. Read two registers using the rs and rt fields of the IR as the register numbers, putting the data into registers A and B.
Register control	Write ALU Write MDR	Write the register file using the rd field of the IR as the register number and the contents of ALUOut as the data. Write the register file using the rt field of the IR as the register number and the contents of the MDR as the data.
Memory	Read PC Read ALU Write ALU	Read memory using the PC as address; write result into IR (and the MDR). Read memory using ALUOut as address; write result into MDR. Write memory using the ALUOut as address; contents of B as the data.
PCWrite control	ALU ALUOut-cond Jump address Seq	Write the output of the ALU into the PC. If the Zero output of the ALU is active, write the PC with the contents of the register ALUOut. Write the PC with the jump address from the instruction. Choose the next microinstruction sequentially.
Sequencing	Fetch Dispatch i	Go to the first microinstruction to begin a new instruction. Dispatch using the ROM specified by i (1 or 2).

Microprogramas

Label	ALU control	SRC1	SRC2	Register control	Memory	PCWrite control	Sequencing
Fetch	Add	PC	4		Read PC	ALU	Seq
	Add	PC	Extshft	Read			Dispatch 1
Mem1	Add	A	Extend				Dispatch 2
LW2					Read ALU		Seq
					Write MDR		Fetch
SW2					Write ALU		Fetch
Rformat1	Func code	A	B				Seq
					Write ALU		Fetch
BEQ1	Subt	A	B			ALUOut-cond	Fetch
JUMP1						Jump address	Fetch

Symbolic Names para os campos das microinstruções

Field Name	Values for Field	Function of Field with Specific Value
ALU	Add Subt. Func code Or	ALU adds ALU subtracts ALU does function code ALU does logical OR
SRC1	PC rs	1st ALU input = PC 1st ALU input = Reg[rs]
SRC2	4 Extend Extend0 Extshft rt	2nd ALU input = 4 2nd ALU input = sign ext. IR[15-0] 2nd ALU input = zero ext. IR[15-0] 2nd ALU input = sign ex., sl IR[15-0] 2nd ALU input = Reg[rt]
destination	rd ALU rt ALU rt Mem	Reg[rd] = ALUout Reg[rt] = ALUout Reg[rt] = Mem
Memory	Read PC Read ALU Write ALU	Read memory using PC Read memory using ALU output Write memory using ALU output
Memory register	IR	IR = Mem
PC write	ALU	PC = ALU
Sequencing	ALUoutCond Seq Fetch Dispatch	IF ALU Zero then PC = ALUout Go to sequential instruction Go to the first microinstruction Dispatch using ROM.

Horizontal vs. Vertical Microprogramming

Unidades de controle microprogramadas:

- **Microprogramação horizontal** (1 bit por ponto a controlar)
- **Microprogramação vertical** (os diferentes campos da microinstrução têm de ser descodificados para gerar os sinais de control)

Horizontal

- + mais control sobre o potencial paralelismo das operações no datapath
- consome muita memória (control store)

Vertical

- + mais fácil de (micro)programar (não muito diferente de programar um processador RISC em assembly)
- um nível extra de descodificação pode tornar o CPU mais lento

ABF_AC1 - MicroprogCU

13

Formato das Microinstruções: unencoded vs. encoded fields

Field Name Width Control Signals Set

wide narrow

ALU Control	4	2	ALUOp
SRC1	2	1	ALUSelA
SRC2	5	3	ALUSelB
ALU Destination	3	2	RegWrite, MemtoReg, RegDst
Memory	4	3	MemRead, MemWrite, IorD
Memory Register	1	1	IRWrite
PCWrite Control	4	3	PCWrite, PCWriteCond, PCSource
Sequencing	3	2	AddrCtl
Total width	26	17	bits

ABF_AC1 - MicroprogCU

14

Microprogramação

- Vantagens:

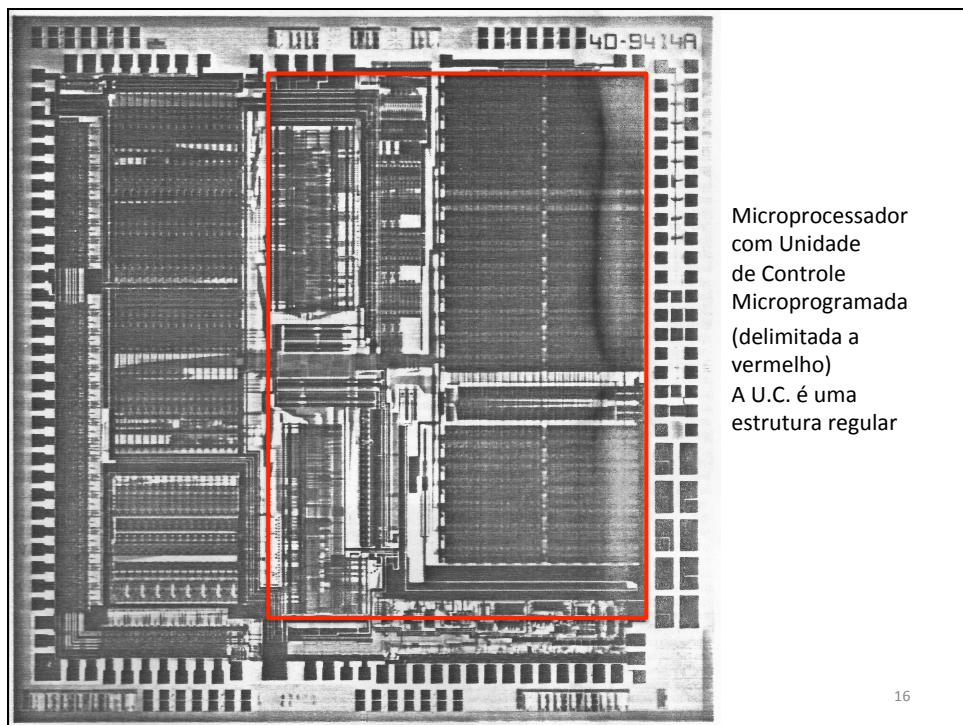
- Permite a implementação de um mesmo reportório de instruções (arquitetura) em recursos de hardware muito diferentes (p.ex. IBM S/360 – arquitetura de 32-bits implementada, simultaneamente, em processadores com níveis de desempenho e custos muito diversos)
- Facilidade de projeto
- Flexibilidade – p.ex. fácil acrescentar instruções ao reportório
- Possível fazer mudanças em fases avançadas do projeto
- Capacidade de implementar instruction sets muito complexos (aumentando a memória de control)
- Generalidade - permite implementar diferentes instruction sets na mesma máquina

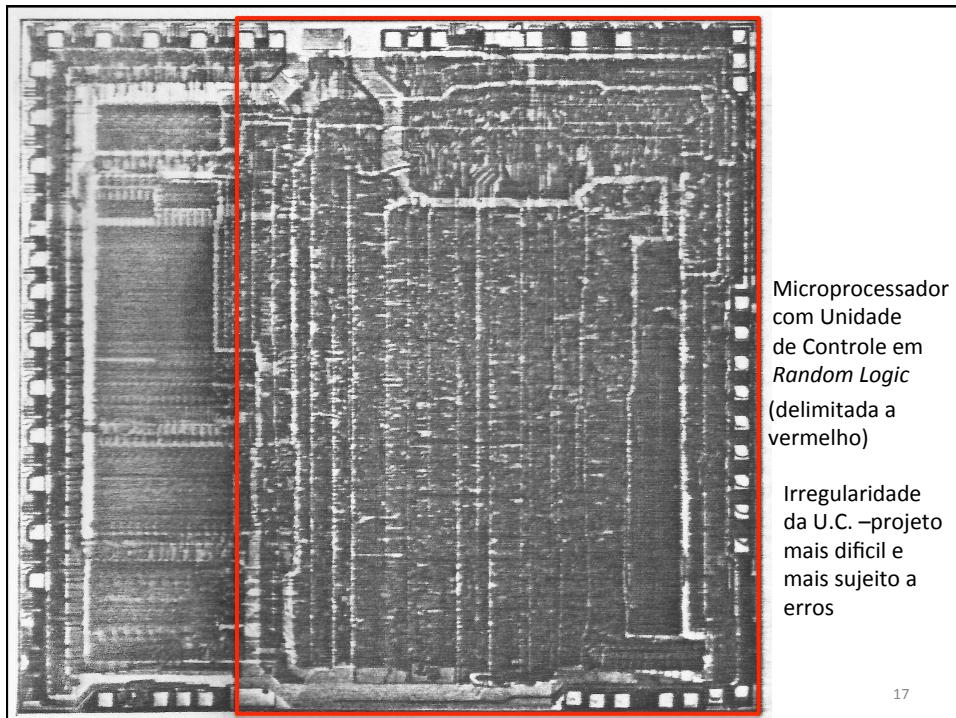
- Desvantagens:

- Processador mais lento

ABF_AC1 - MicroprogCU

15





Processadores Microprogramáveis

- Usar RAM em vez de ROM para a memória de controle
 - Microprogramas escritos na memória de controle quando o computador é ligado.
 - Conteúdo da memória de controle pode ser atualizado (microcode updates)
- IBM S/360 Instruction Set Architecture (ISA): o mesmo repertório de instruções para máquinas muito diferentes em preço e desempenho – datapaths de 8-bit a 32-bit consoante o modelo – o mesmo instruction set executado por “microarquiteturas” muito diferentes
- **Emulação:** executar diferentes Instruction Sets (Arquiteturas) num mesmo hardware
 - Usado pela IBM para executar código máquina de arquiteturas anteriores em computadores S/360, garantindo “software compatibility”
 - A emulação teve um papel importante na aceitação da nova (e incompatível com as anteriores) arquitetura da IBM.

Sumário: Microprogramação e arquiteturas RISC

- Se instruções simples (*RISC*) podem executar com uma elevada frequência de relógio...
- Se é possível escrever compiladores para gerarem microinstruções...
- Se a maioria dos programas usa instruções e modos de endereçagem simples...
- Se o microcódigo residir em RAM e não em ROM para poder corrigir erros...
- Então porque não dispensar a interpretação das instruções por um microprograma e compilar diretamente para a linguagem máquina?

➤ **Arquiteturas RISC**