Organização de Computadores
µArquitetura

Universidade Federal de Uberlândia
Faculdade de Computação
Prof. Dr. rer. nat. Daniel D. Abdala

Na Aula Anterior ...

- Definição de desempenho;
- Mensuração de desempenho;
- Desempenho da UCP e seus fatores;
- Desempenho de instruções;
- Considerações acerca do consumo de energia;
- Desempenho em sistemas multiprocessados;
- Benchmarrking;

Nesta Aula

- O conceito de µarquitetura;
- Organização Monociclo;

Introdução

- Ao estudarmos o conjunto de instruções e as demais “regras do jogo” definidas pela ISA, aprendemos como o processador deve funcionar;
- A μarquitetura ou organização do computador dita como ele é construído;
- Naturalmente utilizamos lógica digital para construir computadores modernos;
- Neste segundo segmento da disciplina construiremos um processador que implementa a ISA que estudamos até agora;

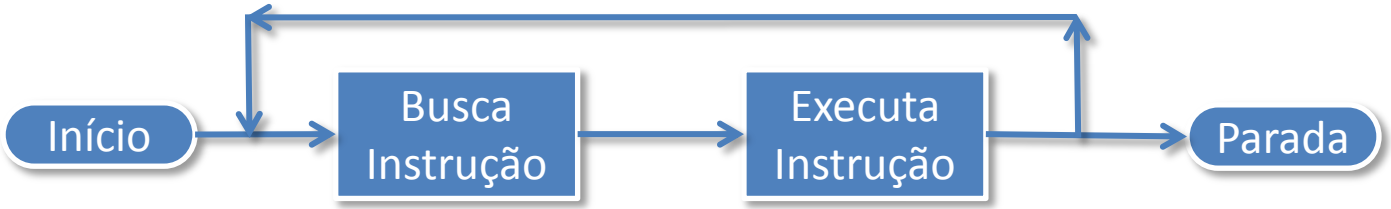
Formas de Organização

- Uma dada ISA pode ser organizada (construída) de diversas formas:
  - Ex: ISA x86 – ambas empresas Intel e AMD produzem processadores compatíveis, mas eles são construídos fundamentalmente de maneira distinta;
  - Uma ISA pode ser implementada com foco em desempenho, minimizar consumo de energia, minimizar lógica computacional, paralelização de instruções, número de ciclos de clock por instrução, etc...

MIPS-Monociclo

- Esta é a forma mais simples de se implementar uma ISA;
- Significa que uma instrução é executada por ciclo de clock;
- Instruções distintas requerem tempos mínimos que variam;
- O Ciclo de clock deve ser definido para acomodar a instrução mais lenta;
- Não é usada atualmente pois é ineficiente;

Ciclo de Execução de Instruções



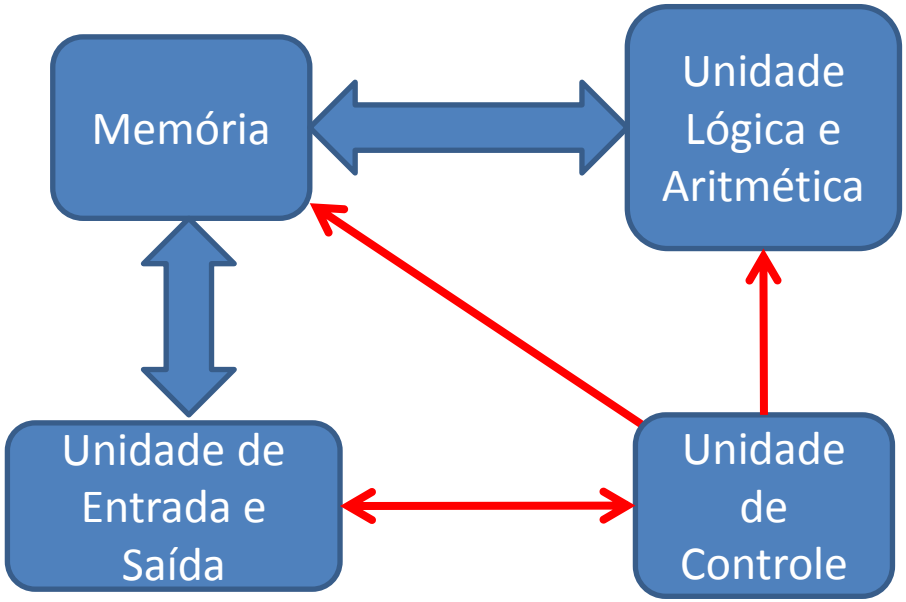
- **INÍCIO** → BOOT/RESET do processador: ao iniciar registrador PC é setado para um endereço pré-determinado, e a instrução neste endereço é buscada para execução;
- **BUSCA INSTRUÇÃO** → Lê a memória (de texto) na posição apontada pelo registrador PC e a envia para o decodificador de instruções;
- **EXECUTA INSTRUÇÃO** → **Decodifica a instrução**, **Busca Operandos** (Registradores, ou memória) e **Executa a Instrução**;
- **PARADA** → o Funcionamento do processador é interrompido por corte de alimentação

Execução de Instruções

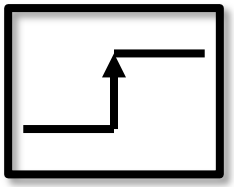
- É subdividido em três passos:
  - **Decodificação**: identifica os campos da instrução e gera os sinais de controle necessários para a execução da instrução;
  - **Busca de Operandos**: Com base na decodificação indentifica quais operandos devem ser buscados (em geral do banco de registradores) e os busca;
  - **Execução**: os sinais de controle chaveiam os circuitos internos para a efetiva execução da instrução decodificada, em geral pela ULA;

Projeto do Subset ISA-MIPS Monociclo

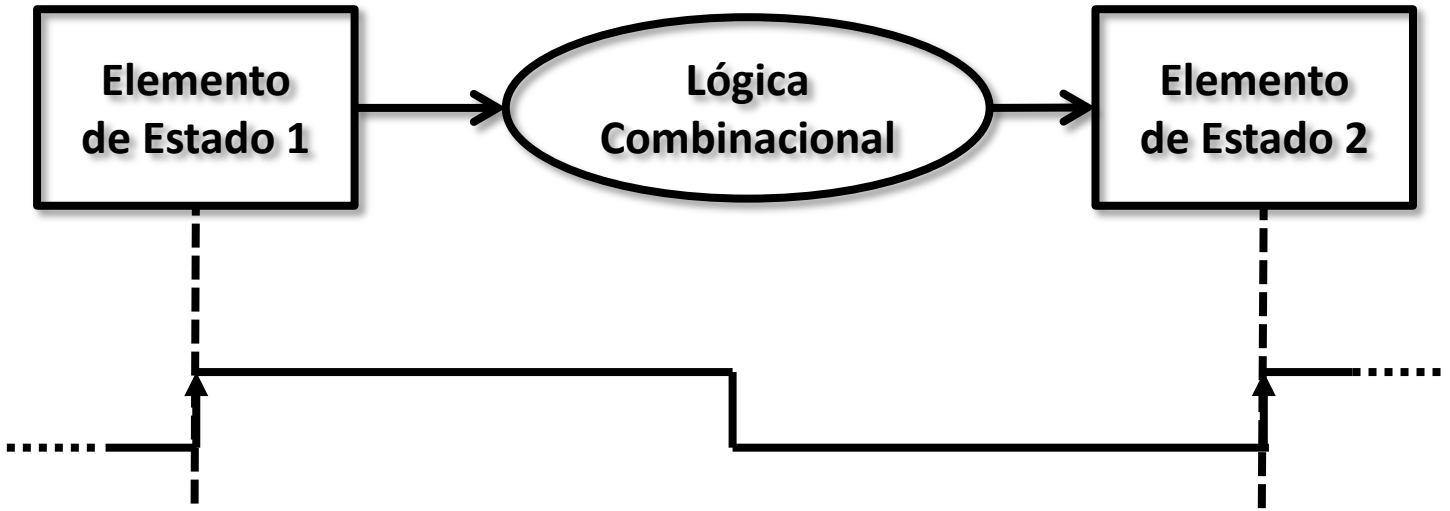
- Hardware projetado para concordar com a ISA;
- Processador pode ser subdividido nas seguintes unidades funcionais:



Metodologia de Clock

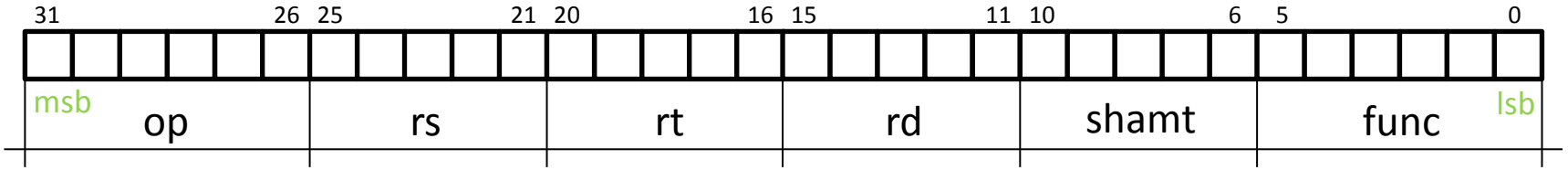


- Define quando sinais podem ser lidos ou escritos;
- Geralmente utiliza-se uma metodologia gatilhada por mudança de estado do clock;

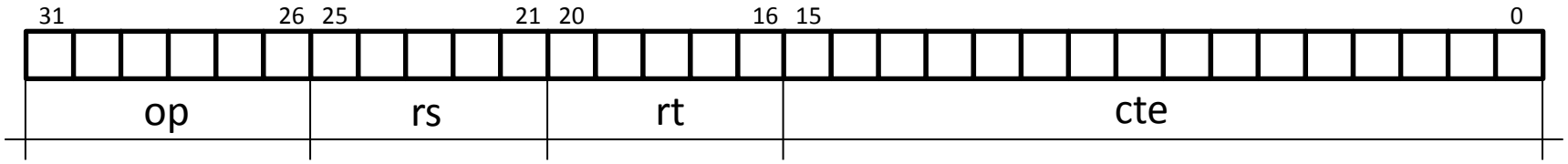


Formatos de Instruções

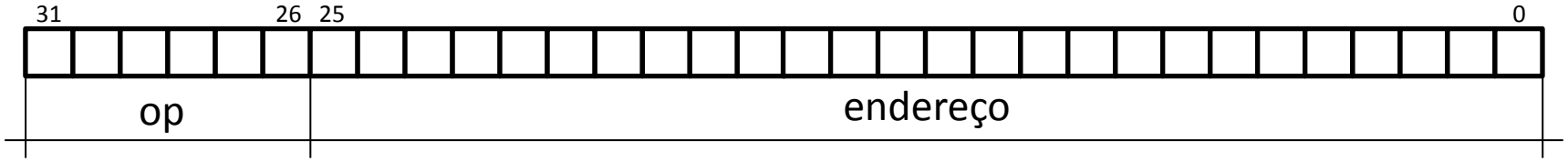
Tipo R



Tipo I



Tipo J

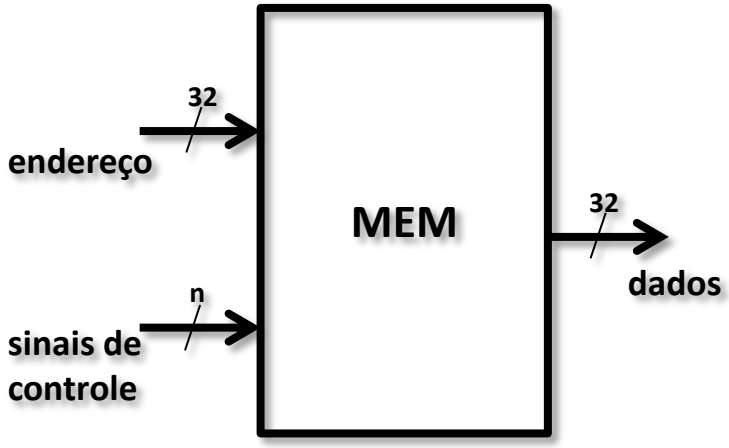


Subconjunto de Instruções

Instrução	Exemplo	Semântica	Tipo
add	add $rd,$rs,$rt	rd ⟵ rs + rt	**R**
sub	sub $rd,$rs,$rt	rd ⟵ rs - rt	**R**
and	and $rd,$rs,$rt	rd ⟵ rs & rt	**R**
or	or $rd,$rs,$rt	rd ⟵ rs | rt	**R**
nor	nor $rd,$rs,$rt	rd ⟵ ~(rs | rt)	**R**
addi	addi $rt,$rs,cte	rt ⟵ rs + cte	**I**
andi	andi $rt,$rs,cte	rt ⟵ rs & cte (16 bits lsb)	**I**
ori	ori $rt,$rs,cte	rt ⟵ rs | cte (16 bits lsb)	**I**
nori	nori $rt,$rs,cte	rt ⟵ ~(rs + cte) (16 bits lsb)	**I**
beq	beq $rt,$rs,LABEL	SE rs=rt pule para LABEL	**R**
slt	slt $rd,$rs,$rt	SE rs<rt SETA rd=01 SENÃO rd=00	**R**
slti	slti $rt,$rs,cte	SE rs<cte SETA rt=01 SENÃO rt=00	**I**
lw	lw rt, cte($rs)	rt ⟵ MEM[rs+cte]	**I**
sw	lw rt, cte($rs)	MEM[rs+cte] ⟵ rt	**I**
j	j LABEL	pule para LABEL	**J**

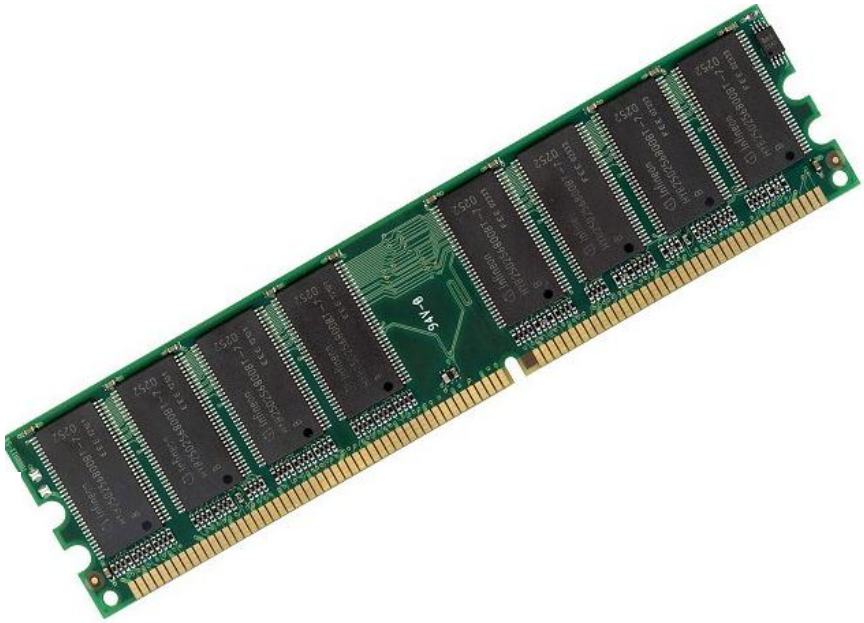
Memória de Programa

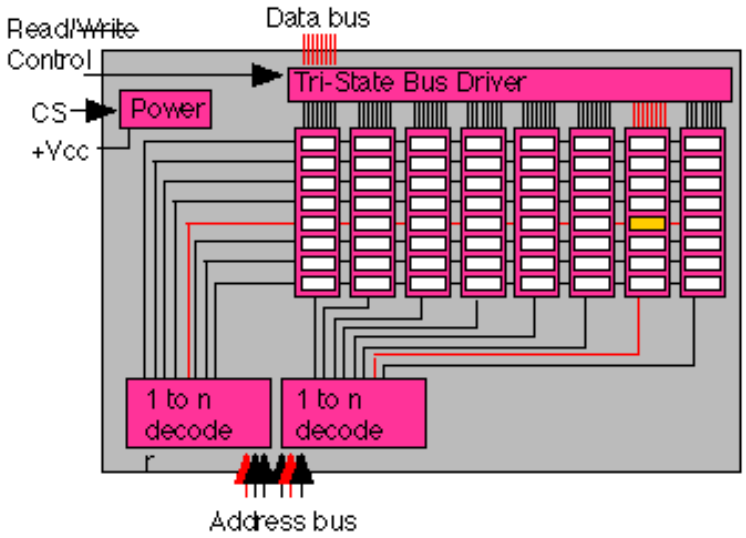
- 2^{30} endereços (palavras)
- 4 bytes por instrução
- 1.073.741.824 instruções
- "texto" do programa deve ser alterado apenas pelo SO;
- Arquitetura Harvard – Programas e dados são armazenados em memórias distintas.



Memória de apenas leitura (perspectiva do processador)

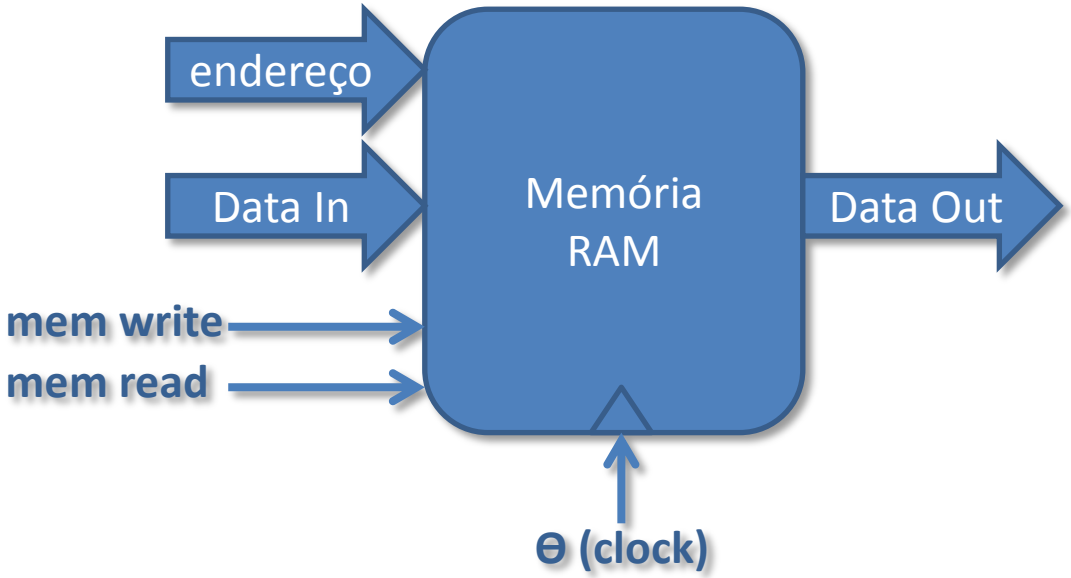
Memória RAM




Read/Write
Control
Data bus
Tri-State Bus Driver
CS
Power
+Vcc
1 to n
decode
1 to n
decode
Address bus

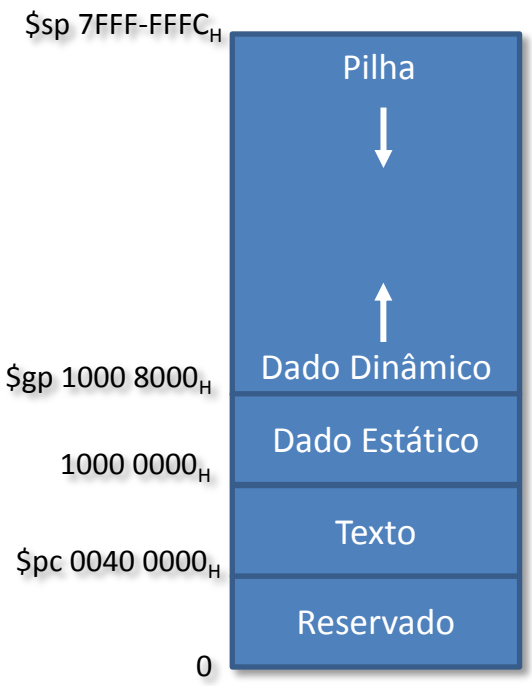
Memória RAM

- Memória principal do sistema;
- Contém dados e programas;
- Volátil;
- Lenta em comparação aos registradores;



Abstração Memória:

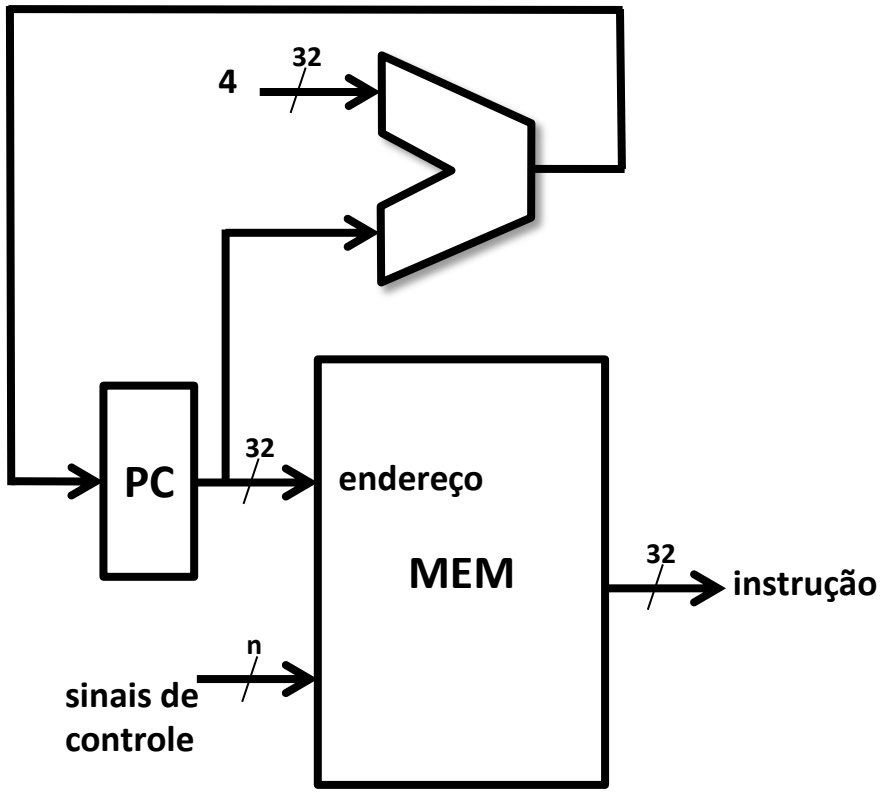
Abstração de Memória



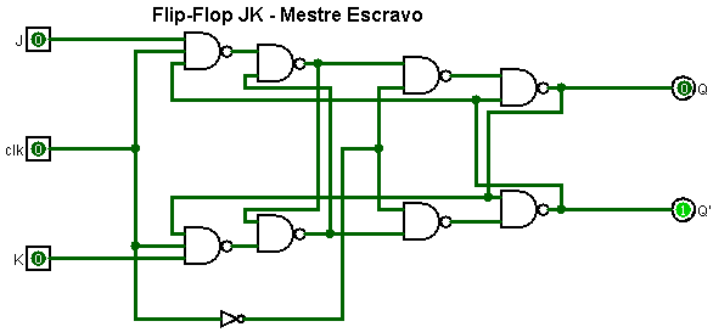
- Do ponto de vista do programador:
  - Array de dados;
  - Endereço identifica uma célula de memória individual;
  - O conteúdo da célula contém efetivamente o dado/instrução;

Busca de Instruções

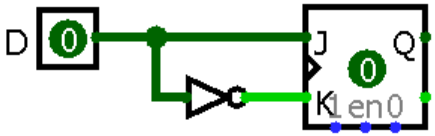
- Qual a prox. Instrução?
- Somam-se 4 bytes pois cada instrução na ISA considerada tem 4 bytes;
- PC sempre aponta para a próxima instrução a ser executada;
- IR contêm a instrução sendo executada.

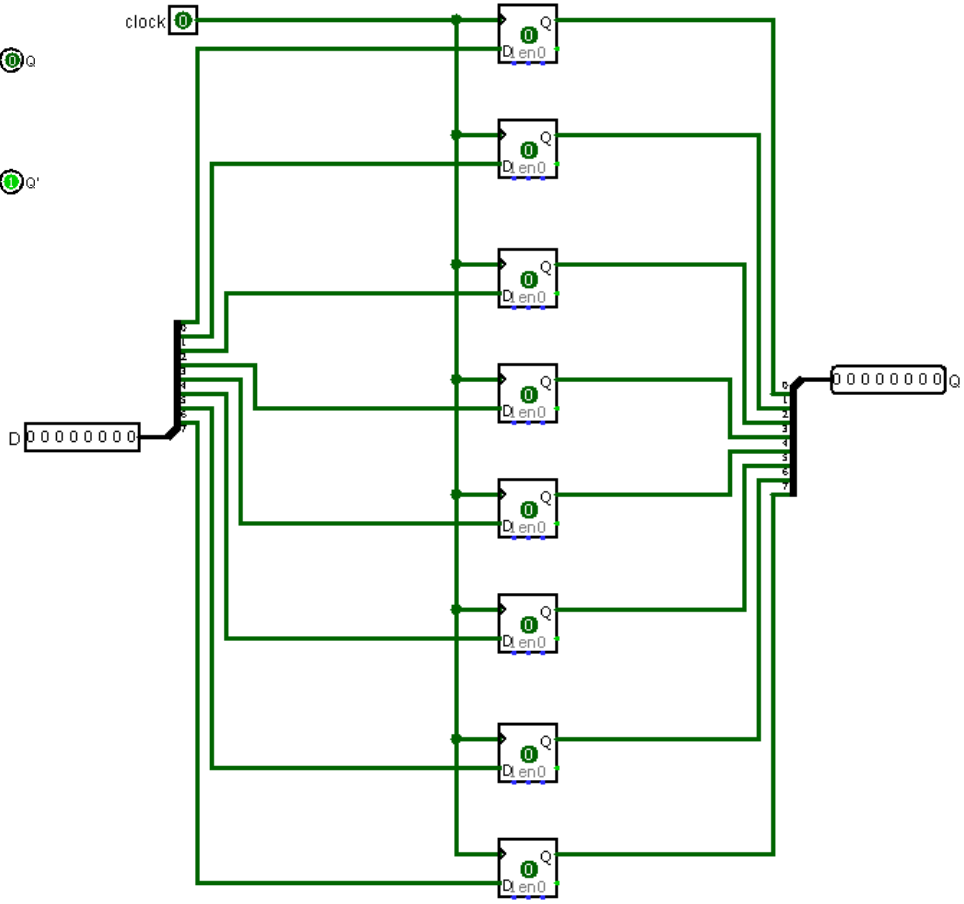


Revisão - Registrador



FLIP-FLOP Tipo D





Registradores

- Apenas 32 registradores?
  - Poucos e rápidos!
- E os registradores $at (1) e $k0,$k1 (26,27)?
  - Montador e SO

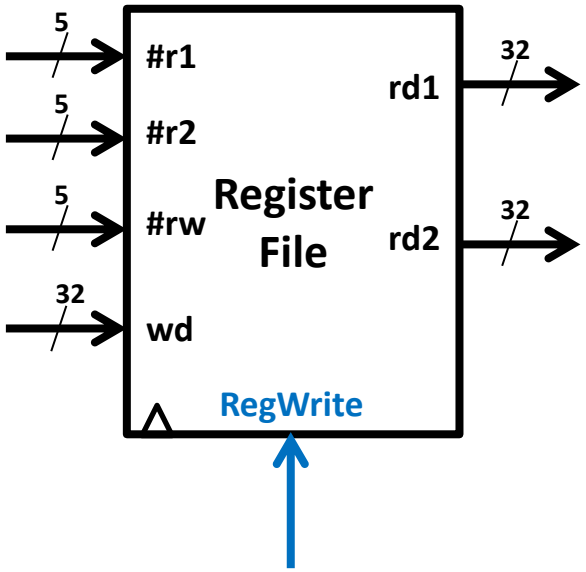
Nome	# do registrador	Uso	Preservado na Chamada?
$zero	0	Constante 0	-
$v0,$v1	2,3	Retorno de funções	Não
$a0-$a3	4-7	Argumentos	Não
$t0-$t7	8-15	Temporários	Não
$s0-$s7	16-23	Salvos	Sim
$t8,$t9	24,25	Mais temporários	Não
$gp	28	Ponteiro global	Sim
$sp	29	Ponteiro de pilha	Sim
$fp	30	Ponteiro de quadro	Sim
$ra	31	Endereço de retorno	Sim

Banco de Registradores

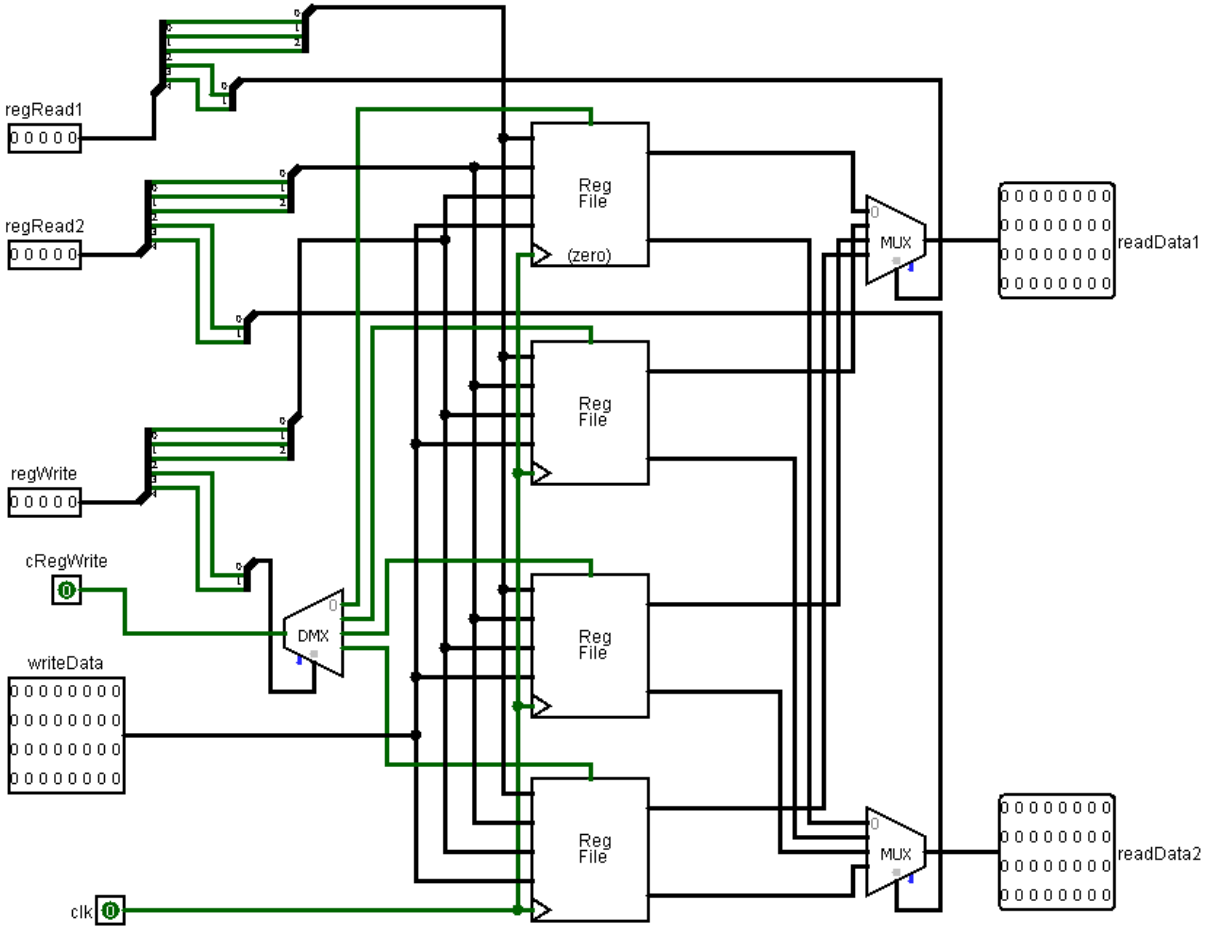
- Register File;
- Parte da CPU que implementa os registradores acessíveis;
- Sempre acessa dois registradores por vez;
- Uma vez endereçado o registrador nas entradas “#r1” e “#r2” o conteúdo dos registradores correspondentes é imediatamente direcionado para as saídas “rd1” e “rd2”;

Register File

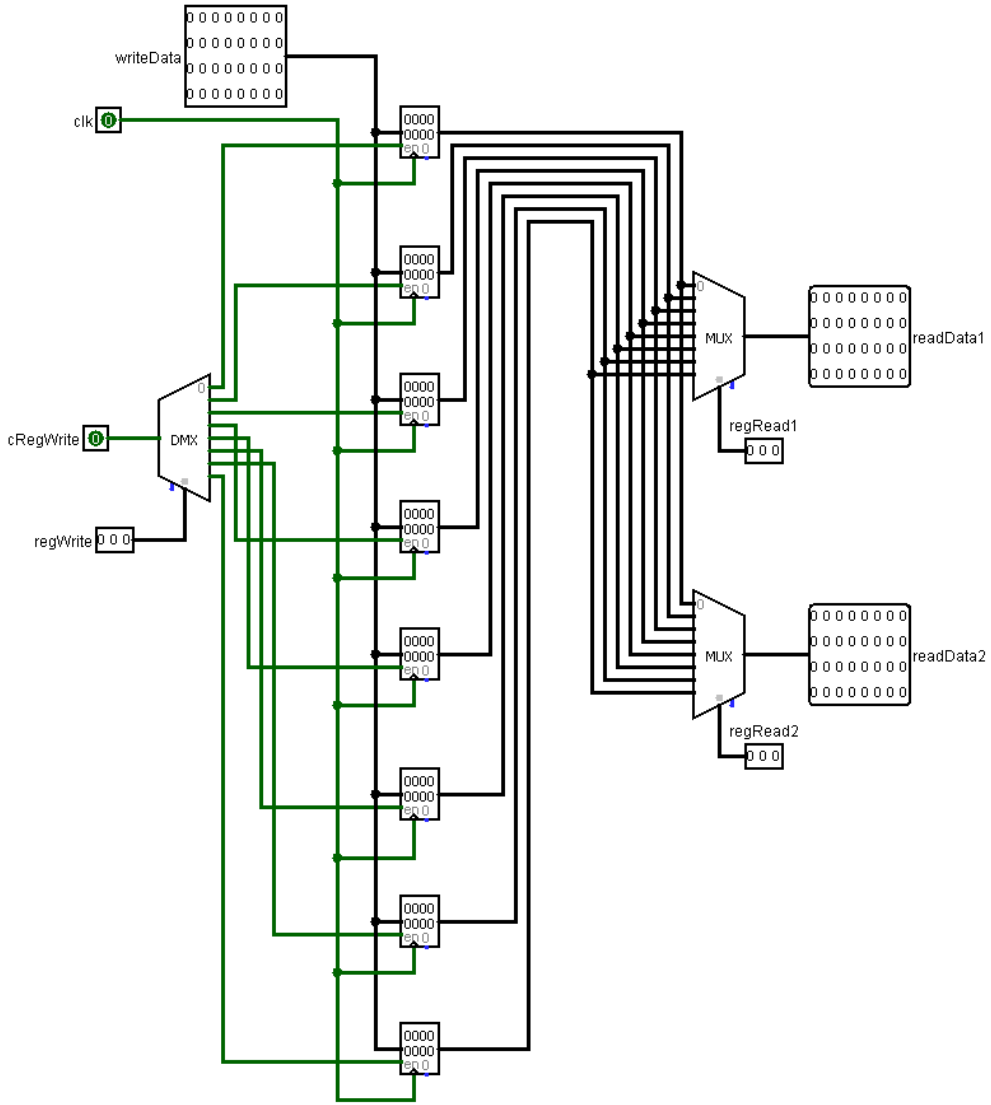
- O File Register prevê a escrita de um registrador por ciclo de clock.
- O endereço do registrador deve ser especificado no campo “#rw” e o dado a ser escrito no campo “wd”;
- O sinal de controle “RegWrite” deve ser colocado em “1” para que a escrita tenha efeito.



Implementação Lógica do RF



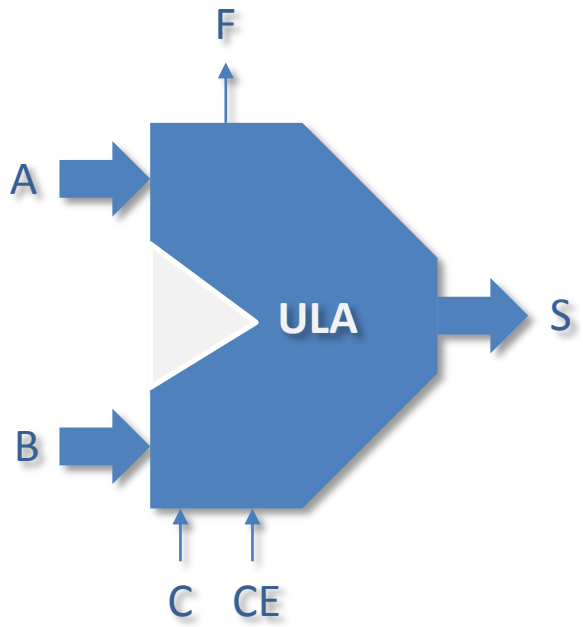
Banco de 8 Registradores



ULA – Unidade Lógica e Aritmética

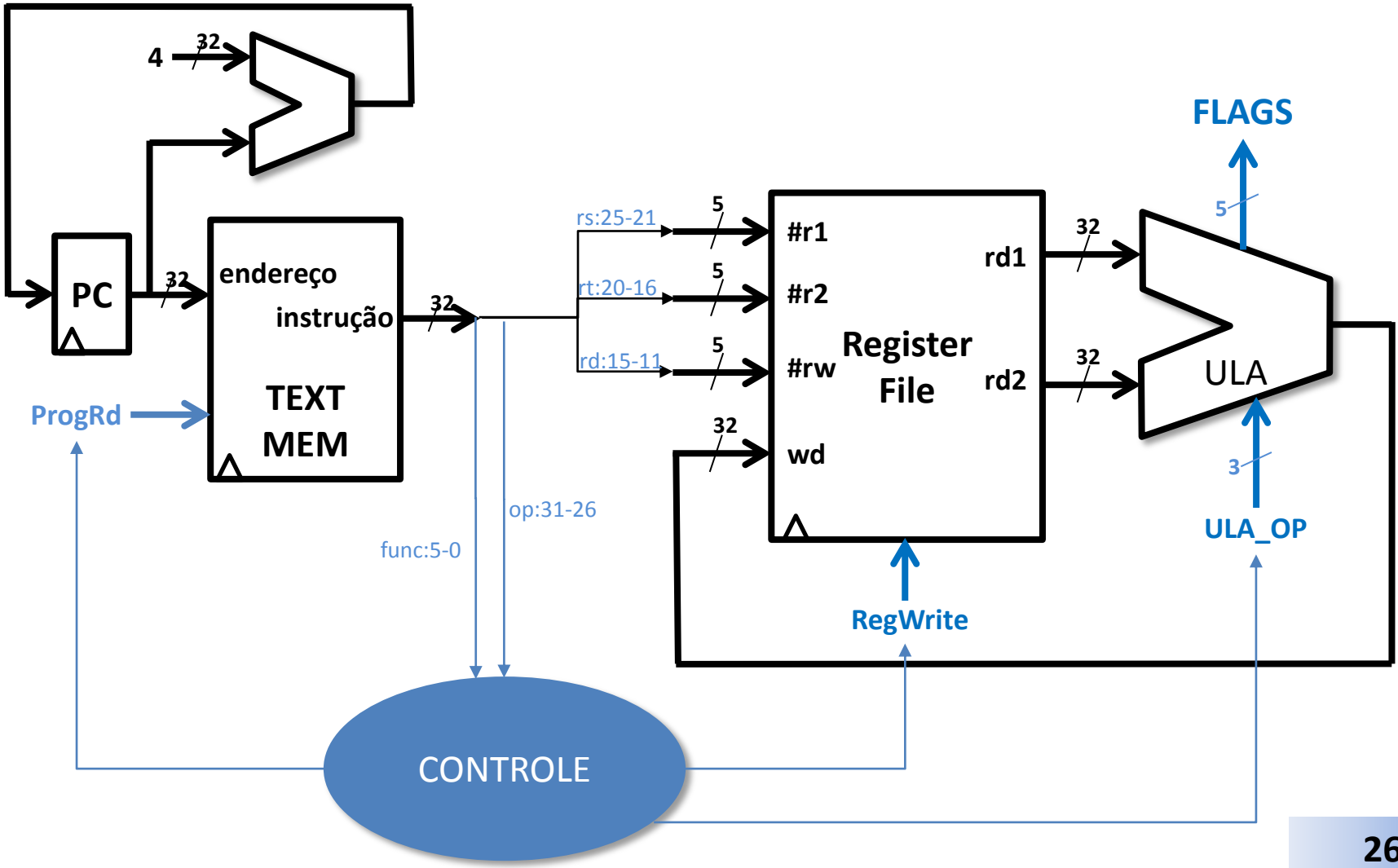
- “Coração” do µprocessador;
- É a coleção de subsistemas digitais que executam boa parte das funções de um processador;

ULA – Visão Geral



- **A,B** – entradas de dados;
- **C** – controle, indica que operação será executada;
- **S** – saída de dados;
- **F** – flags, indica o estado da ULA;
- **CE** – chip enable, habilita o funcionamento da ULA.

Caminho de Dados – Apenas Tipo R



Instruções Suportadas por Esta Organização

ARITMÉTICAS

	MMO	ARGUMENTOS	TIPO	OPCODE	FUNCT
(01)	add	rd, rs, rt	(R)	000000	100000
(02)	sub	rd, rs, rt	(R)	000000	100010

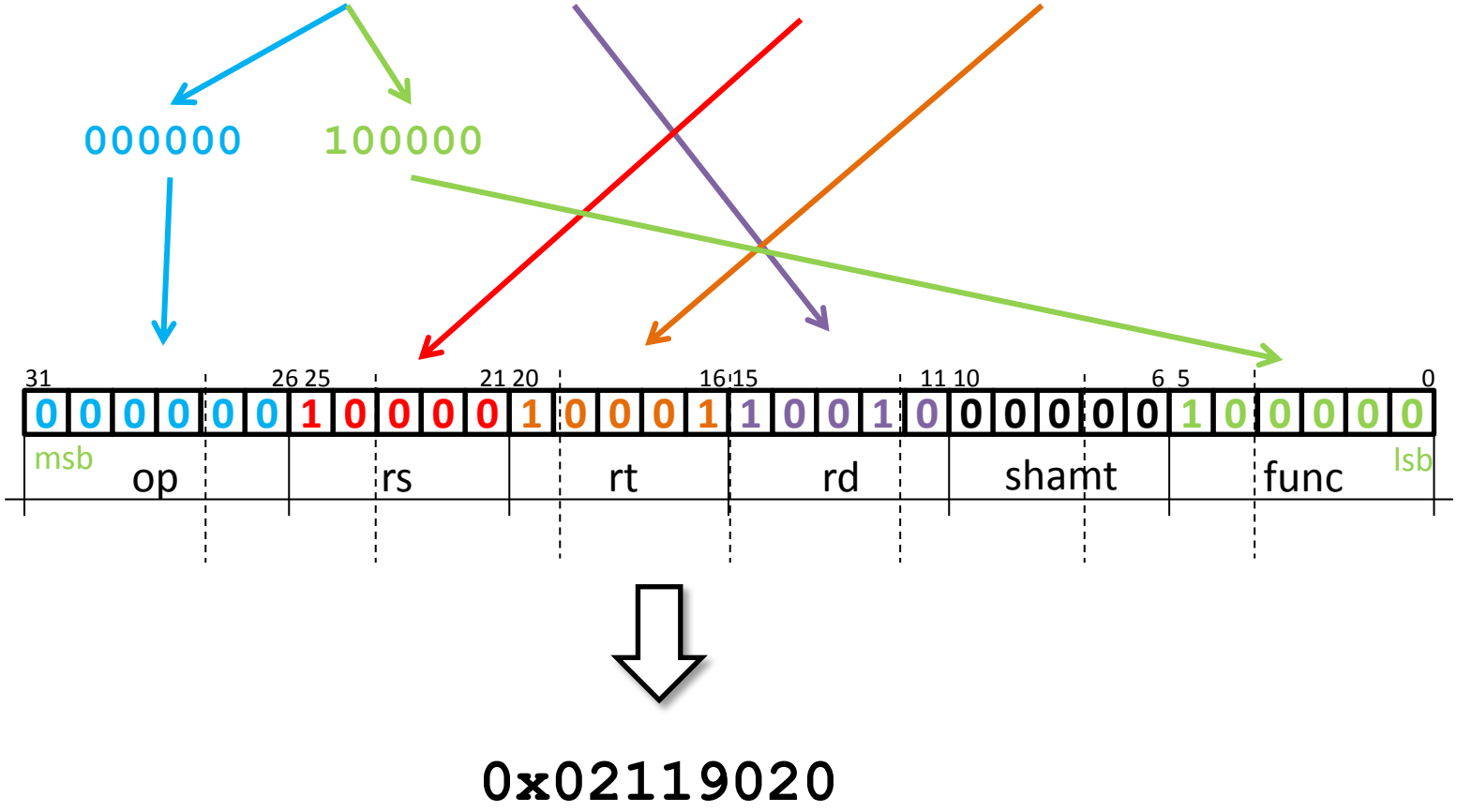
LÓGICAS

	MMO	ARGUMENTOS	TIPO	OPCODE	FUNCT
(03)	and	rd, rs, rt	(R)	000000	100100
(04)	or	rd, rs, rt	(R)	000000	100101
(05)	nor	rd, rs, rt	(R)	000000	100111
(06)	xor	rd, rs, rt	(R)	000000	100110

Instrução	Exemplo	Semântica	Tipo
add	add $rd,$rs,$rt	rd ⟵ rs + rt	**R**
sub	sub $rd,$rs,$rt	rd ⟵ rs - rt	**R**
and	and $rd,$rs,$rt	rd ⟵ rs & rt	**R**
or	or $rd,$rs,$rt	rd ⟵ rs | rt	**R**
nor	nor $rd,$rs,$rt	rd ⟵ ~(rs | rt)	**R**
xor	xor $rd,$rs,$rt	rd ⟵ rs XOR rt	**R**

Executando uma Instrução Tipo R

EX: `add $s2, $s0, $s1`



0x02119020

Caminho de Dados – Apenas Tipo R