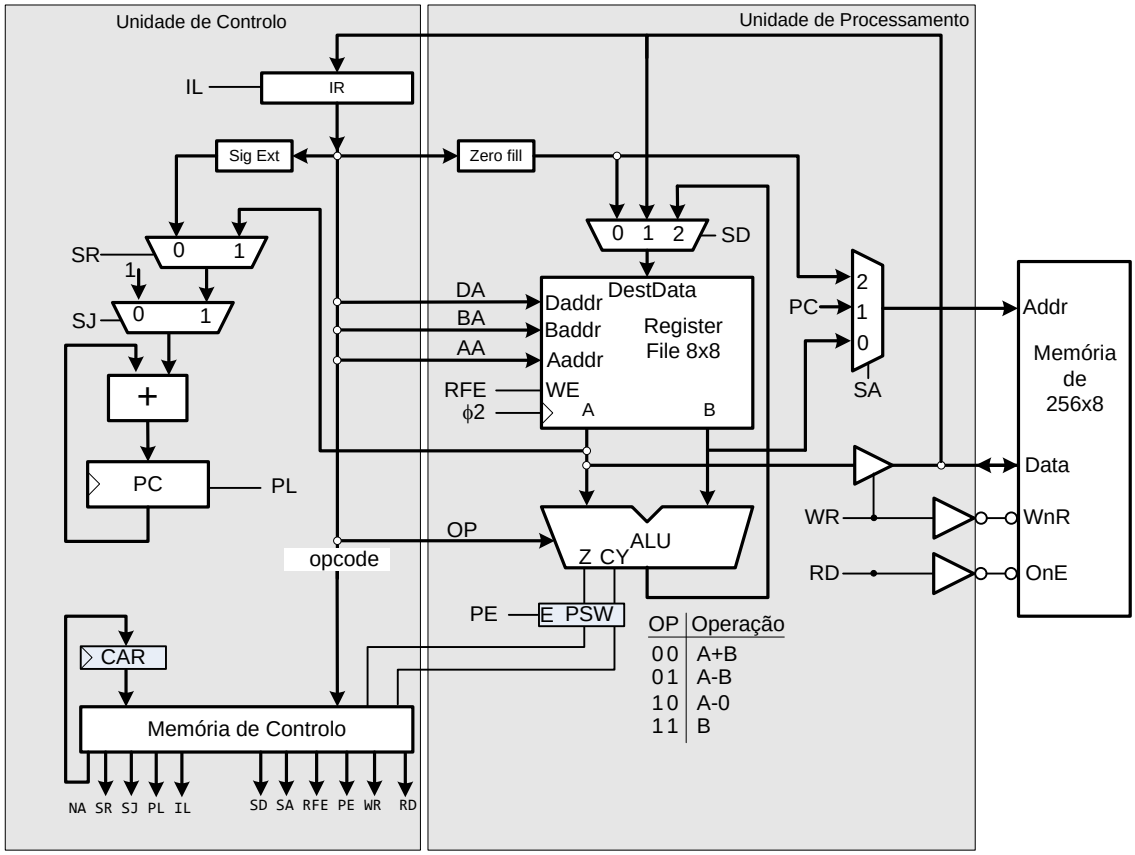
Arquitetura de Computadores

Aula 6 – Processador de múltiplos ciclos; Pipeline

1. Processador de múltiplos ciclos



IR (*Instruction Register*) – Guarda a instrução a ser executada

O processamento da instrução tem duas fases principais: *fetch* e *execute*:

- *Fetch* – lê a instrução da memória e carrega no registo IR
- *Execute* – executa a instrução

O endereço da memória é o PC quando se pretende ler uma nova instrução da memória.
Na fase de execução, o endereço de memória será de acordo com a instrução a executar.

O descodificador de instruções passa s ser sequencial, uma vez que necessitamos mais do que um ciclo para executar uma instrução.

2. Palavra de controlo
   NA, SR, SJ, PL, IL, MC, SD, SA, RFE, PE WR, RD

3. Descodificação da instrução

   Circuito sequencial:

   Entradas: instrução (opcode, flags, endereço memória de controlo)

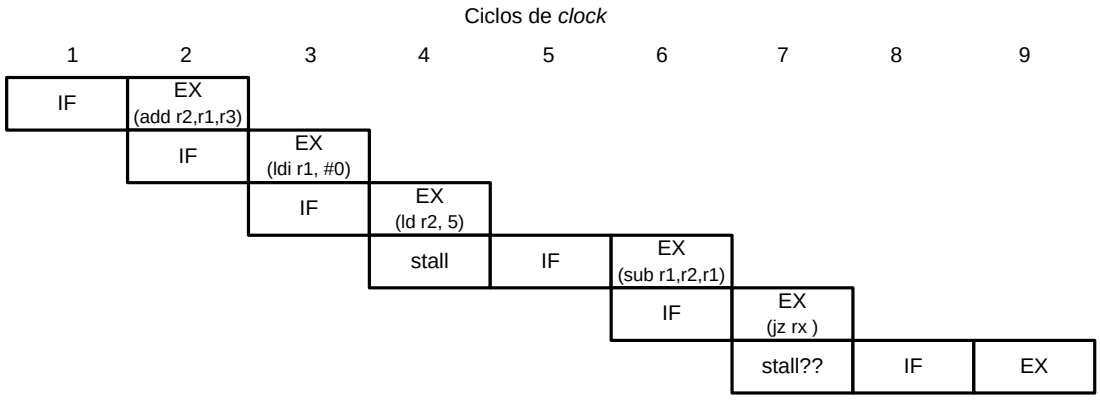
   Saídas: palavra de controlo

CAR	instr	Z	C	NA	SR	SJ	PL	IL	SD	SA	RFE	PE	WR	RD
0	- - - -	-	-	1	-	0	1	1	--	01	0	0	0	1
1	LD rx, direct5	-	-	0	-	-	0	0	01	10	1	0	0	1
1	LD rx, [ry]	-	-	0	-	-	0	0	01	00	1	0	0	1
1	ST rx, [ry]	-	-	0	-	-	0	0	--	00	0	0	1	0
1	LDi, Rx, #const5	-	-	0	-	-	0	0	00	--	1	0	0	0
1	ADD rz, rx, ry	-	-	0	-	-	0	0	10	--	1	1	0	0
1	SUB rz, rx, ry	-	-	0	-	-	0	0	10	--	1	1	0	0
1	DEC rx, rx	-	-	0	-	-	0	0	10	--	1	1	0	0
1	JMP offset7	-	-	0	0	1	1	0	--	--	0	0	0	0
1	JNZ offset7	0	-	0	0	1	1	0	--	--	0	0	0	0
1		1	-	0	-	-	0	0	--	--	0	0	0	0
1	JC offset7	-	0	0	-	-	0	0	--	--	0	0	0	0
1		-	1	0	0	1	1	0	--	--	0	0	0	0
1	JZ rx	-	-	0	1	1	1	0	--	--	0	0	0	0

4. Processador com *pipeline*

Para melhorar o desempenho, usamos a técnica de *pipeline*.

Consideremos o exemplo simples do processador de 2 ciclos anterior.



Desempenho sem *pipeline*: 6 ciclos x 2 = 12 ciclos de *clock*

Desempenho ideal com pipeline: 7 ciclos (nº instruções + nº ciclos encher pipeline)

Desempenho com *pipeline* com *stalls*: 9 ciclos