irmovl : copia o valor da instrução (V) para registo rB

3	0	8	rB	V

↳ Era escusado
Conteúdo, formato da instrução - variante facilita
" " " seguinte
Já temos custo e vantagens do descodificação
estas questões dizem-se das implementações

cada instrução (1 ciclo) é fase sequenciais

rmmovl rA, D(rB) : 4 0 rA rB | D

Endereço (R[rB]+D) onde queremos armazenar VALOR de rA

Execução:
VALE = VALB + VALC

POP rA (semelhante a) | 8 | 0 | rA | 8 |
(load)
tem as mesmas fases (load da stack)
VALA = R[%.ESP] não R[rA]
VALB = R[%.ESP]

Precisamos do ESP e do ESP+4

Sempre que queremos ler temos que dizer a onde. Poupamos fios se só disséssemos o que precisamos (tb é preciso VALB)

jump: não há registos
incondicional: o próximo valor do PC é a constante da instrução
condicional: consultar os valores da condição do código anterior para decidir se usa VALC ou VALP
Decisão a 0 usa o C (PC = valC : VALP)

call:

PC = valC (próxima instrução é valC)
colocamos o endereço da instrução seguinte (VALP) → Return

Return: vamos buscar o valor à memória → guardámos no contador
PC: valor que vem do pilha | (VALP = PC + 1) só ocupa 1 byte
" indicado pelo ESP

Aula 10-11-2012

I Datapath sequencial
Fluxo dos dados
Nunca foi implementado
Um ciclo para cada fase
Não explora o paralelismo
Todas as escritas em memória têm organização sequencial

Na leitura: o valor está sempre a ser armazenado
Escrita é trigada, controlada pelo relógio

Se fosse 64 bits: Registos de 64 bits
PC de 64 bits → ALU de capacidade para inteiros de 64 bits
Ligações de 64 bits (dobro da capacidade)
VALA, VALB ...