Unidade de controlo

- A unidade de controlo deve gerar os sinais (identificados a vermelho) para:
  1) controlar a escrita e/ou a leitura em elementos de estado: **banco de registos** e **memória de dados**
  2) definir a operação dos elementos combinatórios: **ALU** e ***multiplexers***
- A operação na ALU é definida com 3 bits (ALU Control):

ALU operation	ALU Control
AND	000
OR	001
ADD	010
SUB	110
SLT	111

Unidade de controlo

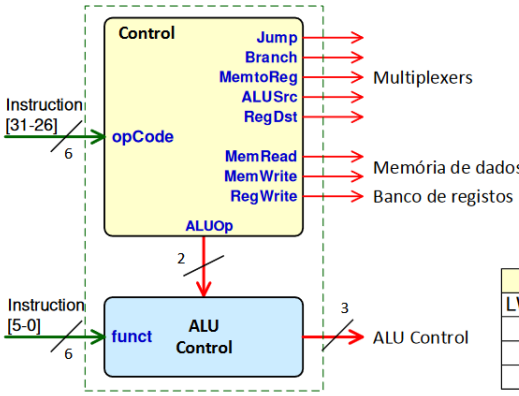
- Alguns dos elementos de estado do *datapath* são acedidos em todos os ciclos de relógio (PC e memória de instruções)
  - Nestes casos não há necessidade de explicitar um sinal de controlo
- Outros elementos de estado podem ser lidos ou escritos dependendo da instrução que estiver a ser executada (memória de dados e banco de registos)
  - Para estes é necessário explicitar os respetivos sinais de controlo
- Nos elementos de estado:
  - a **escrita** é sempre realizada de forma síncrona
  - a **leitura** é sempre realizada de forma assíncrona

Unidade de controlo

- Todas as instruções (exceto o "j") usam a ALU:
  - **LW e SW** – para calcular o endereço da memória externa (soma)
  - **Branch if equal** / **not equal** – para determinar se os operandos são iguais ou diferentes (subtração)
  - **Aritméticas e lógicas** – para efetuar a respetiva operação
- A operação a realizar na ALU depende:
  - dos campos **opcode** e **funct** nas instruções aritméticas e lógicas de tipo R (opcode=0):
    ALUControl = f(opcode, funct)
  - do campo **opcode** nas restantes instruções:
    ALUControl = f(opcode)

Unidade de controlo

- A unidade de controlo pode ser sub-dividida em duas: 1) controlo de multiplexers e elementos de estado; 2) controlo da ALU



ALU Control	ALU operation
000	AND
001	OR
010	ADD
110	SUB
111	SLT

Instruction	ALUOp	ALU operation
LW, SW, ADDI	**00**	ADD
BEQ	**01**	SUB
R-Type	**10**	Depends on "**funct**"
SLTI	**11**	SLT

- A operação da ALU é definida em conjunto com a unidade de controlo principal, em função dos campos "opcode" e "funct"

Análise do funcionamento do *datapath*

- A execução de qualquer uma das instruções suportadas ocorre no intervalo de tempo correspondente a um único ciclo de relógio: tem início numa transição ativa do relógio e termina na transição ativa seguinte
- Para simplificar a análise podemos, no entanto, considerar que a utilização dos vários elementos operativos ocorre em sequência e decorre ao longo de um conjunto de operações
- A sequência de operações culmina com:
  - escrita no Banco de Registos: instruções tipo R, LW, ADDI, SLTI
  - escrita na Memória de Dados: SW
- O *Program Counter* é sempre atualizado com:
  - endereço-alvo da instrução BEQ, se os registos forem iguais (*branch taken*), ou PC+4 se forem diferentes (*branch not taken*)
  - endereço-alvo da instrução J
  - PC+4 nas restantes instruções

Análise do funcionamento do *datapath* – operações

- *Fetch* de uma instrução e cálculo do endereço da próxima instrução
- Leitura de dois registos do Banco de Registos
- A ALU opera sobre dois valores (a origem do segundo operando depende do tipo de instrução que estiver a ser executada)
- O resultado da operação efetuada na ALU:
  - é escrito no Banco de Registos (**R-Type**, **addi** e **slti**)
  - é usado como endereço para escrever na memória de dados (**sw**)
  - é usado como endereço para fazer uma leitura da memória de dados (**lw**) - o valor lido da memória de dados é depois escrito no Banco de Registos
  - é usado para decidir qual o próximo valor do PC (**beq** / **bne**): BTA ou PC+4

Funcionamento do *datapath* nas instruções tipo R

- A instrução é lida e é calculado o valor de PC+4
- São lidos dois registos e a unidade de controlo determina, a partir do *opcode* (**bits 31-26**), o estado dos sinais de controlo
- A ALU opera sobre os dados lidos dos dois registos, de acordo com a função codificada no campo *funct* (**bits 5-0**) da instrução
- O resultado produzido pela ALU será escrito no registo especificado nos **bits 15-11** da instrução ("**rd**"), na próxima transição ativa do relógio

Funcionamento do *datapath* na instrução LW

- A instrução é lida e é calculado o valor de PC+4.
- É lido um registo e a unidade de controlo determina, a partir do *opcode*, o estado dos sinais de controlo.
- A ALU soma o valor lido do registo especificado nos **bits 25-21** ("**rs**") com os 16 bits (estendidos com sinal para 32) do campo *offset* da instrução (**bits15-0**).
- O resultado produzido pela ALU constitui o endereço de acesso à memória de dados. A memória é lida nesse endereço (leitura assíncrona).
- A *word* lida da memória será escrita no registo especificado nos **bits 20-16** da instrução ("**rt**"), na próxima transição ativa do relógio.

Funcionamento do *datapath* na instrução BEQ

- A instrução é lida e é calculado o valor de PC+4
- São lidos dois registos e é determinado o estado dos sinais de controlo. Os 16 LSbits da instrução (sign extended x 4) são somados a PC+4 (BTA)
- A ALU faz a subtração dos dois valores lidos dos registos
- A saída "Zero" da ALU é utilizada para decidir qual o próximo valor do PC, que será atualizado na próxima transição ativa do relógio

Funcionamento do *datapath* na instrução J

- A instrução é lida e é calculado o valor de PC+4
- São determinados os sinais de controlo. O endereço alvo é obtido a partir dos 26 LSbits da instrução multiplicados por 4 (*shift left 2*) concatenados com os 4 bits mais significativos do PC+4