Arquitetura de Computadores

Turma LI21N/LT21N

AULA 12
Arquitectura P16

TÓPICOS A ABORDAR

- *Características principais*
- *Instruction Set Architecture*
- *Registos, formato das instruções e instruções de processamento de dados*

Ano Lectivo 2019/2020

2º Semestre

Prof. Jorge Fonseca

P16 - ESPECIFICAÇÕES

O P16 é um processador de 16 bits com a seguinte especificação:

- Arquitectura LOAD/STORE;
- Banco de registos (*Register File*) com 16 registos de 16 bits;
- Espaço de memória para código e dados 64K*8 com possibilidade de acesso a 8 ou 16 bits;
- ISA, instruções têm tamanho fixo e ocupam uma única palavra de memória (16 bits);
- Suporte à implementação de rotinas;
- Suporte à implementação de estrutura de dados *stack.*
- Acesso a memória e periféricos no mesmo espaço de endereçamento.
- Suporte a interrupções externas
- Sincronização na transferência de dados com dispositivos externos;
- Partilha de barramentos com outros dispositivos

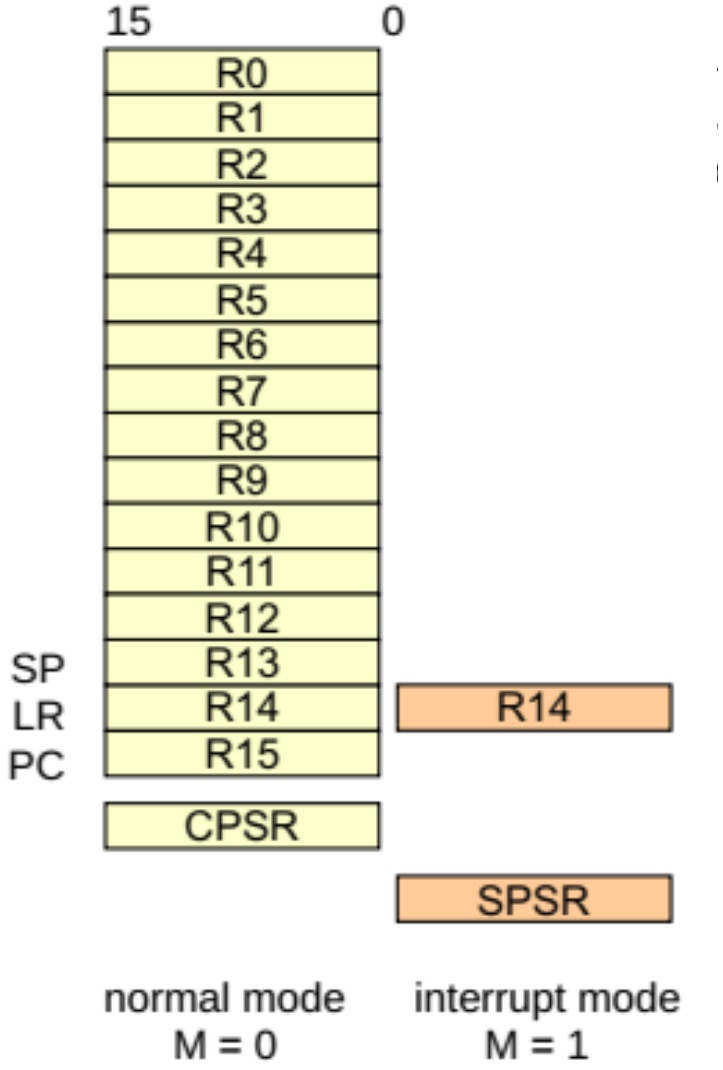
O banco de registos é constituído por 16 registos de 16 bits de R0 a R15, dividido em dois blocos de 8 registos.

Um dos blocos, denominado por bloco baixo é constituído pelos registos de R0 a R7 e o bloco alto, constituído pelos registos de R8 a R15.

Os registos do bloco baixo são de uso genérico e acessível por qualquer das instruções do processador que tenha um registo como parâmetro.

Os registos do bloco alto só são acessíveis por algumas instruções, não podendo ser utilizados por todas as instruções de forma indiscriminada.

No bloco alto, estão incluídos três registos de uso específico: R13/SP (*Stack pointe*r), R14/LR (*Link register)* e o R15/PC (*Program counter*).



O P16 tem uma arquitectura RISC - LOAD/STORE, formada por três grupos de instruções:

- transferência entre registos e memória,
- processamento de dados
- controlo de fluxo.

O acesso à memória pode realizar-se com os seguintes modos de endereçamento:

- indirecto;
- baseado e indexado.

No acesso indirecto, o endereço é especificado a 6 bits sem sinal relativo ao PC. Só disponível para leitura e o acesso é sempre realizado a 16 bits (*Word*).

No acesso baseado e indexado, o endereço é calculado pela soma de :

- um registo base e um registo índice
- um registo base e um índice constante de 4 bits.

Nas instruções de transferência, de e para memória, caso se pretenda realizar o acesso ao *byte*, acrescenta-se a letra **b** à direita da mnemónica.

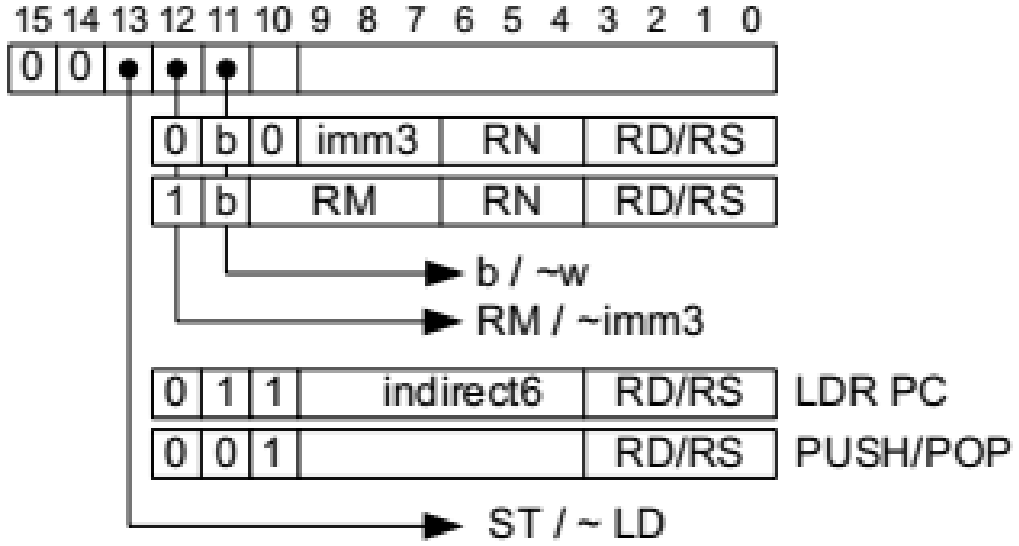
No caso da leitura de um *byte* de memória para registo são adicionados oito zeros na parte alta do *byte* lido. No caso da escrita de um *byte* em memória, o byte a ser escrito corresponde aos 8 bits de menor peso do registo fonte (**RS**) e o conteúdo da memória alterado será o *byte* de menor peso se o endereço for par ou o de maior peso se o endereço for ímpar.

O P16 implementa em memória uma estrutura de dados tipo *stack*. Para suporte a esta estrutura, o P16 utiliza o registo R13 como registo s*tack* p*ointer (SP*), e põe disponíveis duas instruções:

- instrução **push -** para empilhar o valor de um dos registos e
- instrução **pop -** para desempilhar o valor para um dos registos

O *stack* aumenta em memória no sentido dos endereços menores. Os valores empilhados são sempre palavras de 16 *bits*.

O conjunto das instruções de transferência entre registos e memória obedece aos seguintes formatos:



RD/RS - registo destino/fonte (R0 a R15)
RM - registo índice (R0 a R15)
RN - registo base (R0 a R7)
indirect6 - constante de 6 bits sem sinal
imm3 - índice de 3 bits sem sinal
b - indica se a transferência é de um byte ou de uma word

ldr	rd, [pc, <imm6>]	0	0	0	0	1	1	imm6			
pop	rd	0	0	0	0	0	1				rd
push	rs	0	0	1	0	0	1				rs

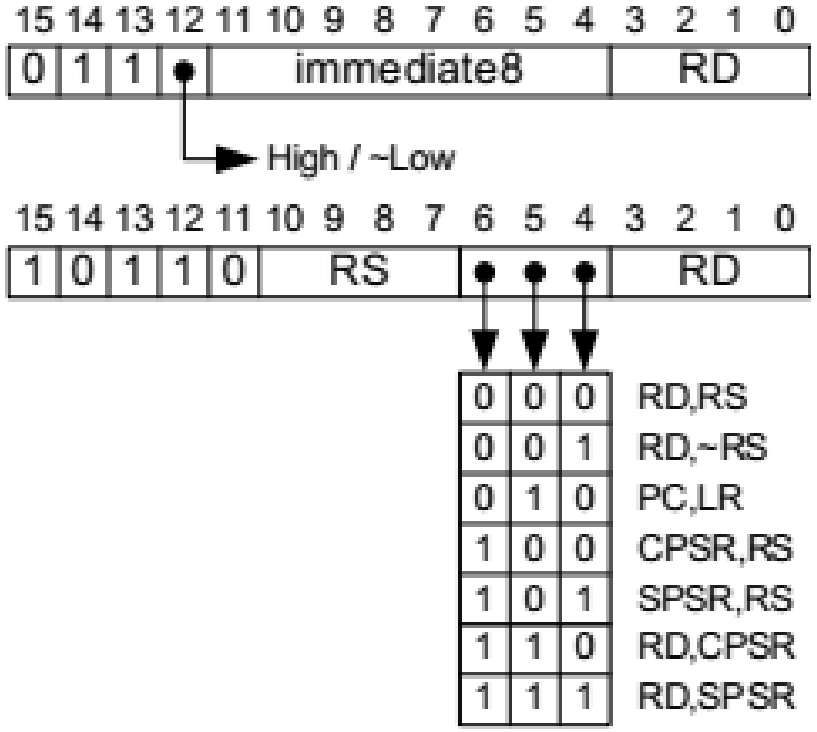
ldr	rd, [rn, <imm3>]	0	0	0	0	0	0	imm3	rn	rd
ldrb	rd, [rn, <imm3>]	0	0	0	0	1	0	imm3	rn	rd
ldr	rd, [rn, rm]	0	0	0	1	0	rm		rn	rd
ldrb	rd, [rn, rm]	0	0	0	1	1	rm		rn	rd

str	rs, [rn, <imm3>]	0	0	1	0	0	0	imm3	rn	rs
strb	rs, [rn, <imm3>]	0	0	1	0	1	0	imm3	rn	rs
str	rs, [rn, rm]	0	0	1	1	0	rm		rn	rs
strb	rs, [rn, rm]	0	0	1	1	1	rm		rn	rs

Instruções para transferência entre registos.

Neste conjunto de instruções estão incluídas, para além das de transferência entre registos do *register file*, instruções com valores imediatos que permitem iniciar um registo com uma constante e instruções que permitem a transferência entre registos do *register file* e o PSR.

As instruções de transferência entre registos obedecem ao seguinte formato:



RD - registo destino/fonte (R0 a R15)

RS - registo fonte (R0 a R15)

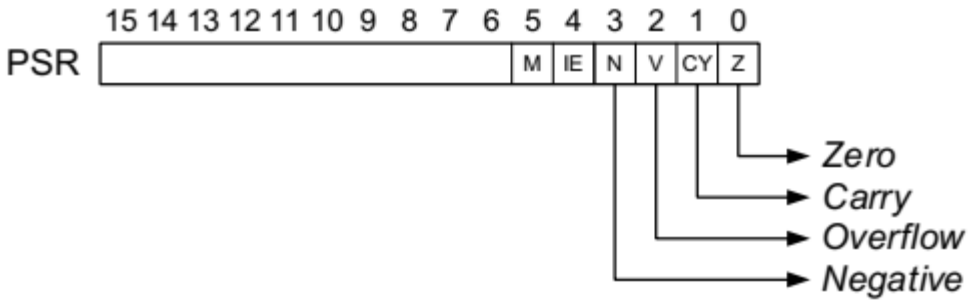
imm8 - constante de 8 bits a ser carregado no registo destino.

mov	rd, rs	1	0	1	1	0	rs		0	0	0	rd
mvn/not	rd, rs	1	0	1	1	0	rs		0	0	1	rd
movs	pc, lr	1	0	1	1	0			0	1	0	
msr	cpsr, rs	1	0	1	1	0	rs		1	0	0	
msr	spsr, rs	1	0	1	1	0	rs		1	0	1	
mrs	rd, cpsr	1	0	1	1	0			1	1	0	rd
mrs	rd, spsr	1	0	1	1	0			1	1	1	rd

mov	rd, <imm8>	0	1	1	0	imm8	rd
movt	rd, <imm8>	0	1	1	1	imm8	rd

As instruções de processamento de dados podem especificar três registos, ou dois registos e uma constante, e incluem instruções de deslocamento.

Estas instruções afectam quatro *flags* que são guardadas no registo denominado por *Processor Status Register (**PSR***). Este registo tem o seguinte formato:



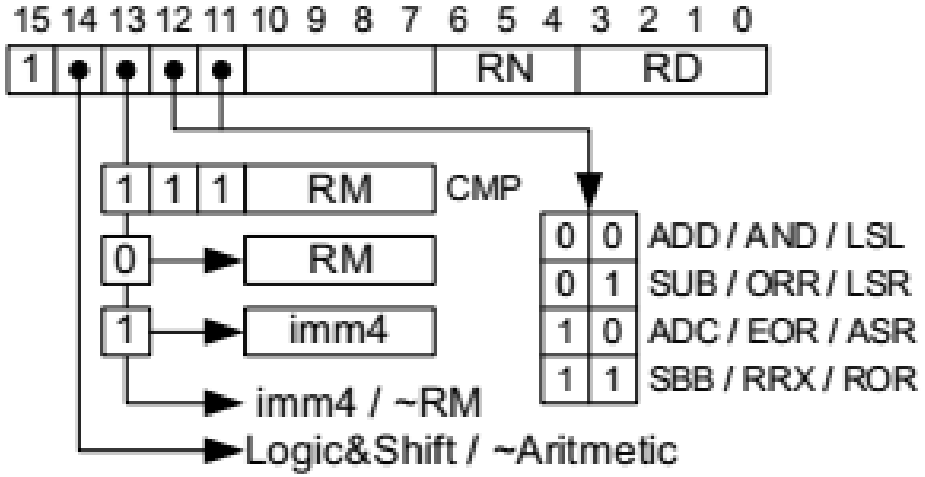
A *flag Zero (Z)* toma o valor lógico um, quando ao realizar uma operação lógica ou aritmética o valor resultante seja igual a zero.

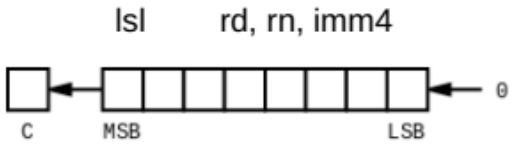
A *flag Carry* (C) fica a um, quando após uma operação aritmética resulte transporte para o dígito mais significativo, ou nas operações de deslocamento o último *bit* deslocado seja 1.

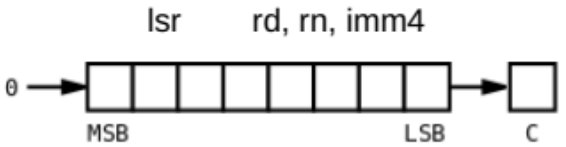
A *flag Overflow (V)* fica ao valor lógico um, quando ao realizar uma subtracção ou uma soma o resultado excede o domínio, entendidos os valores como inteiros com sinal em código dos complementos para dois.

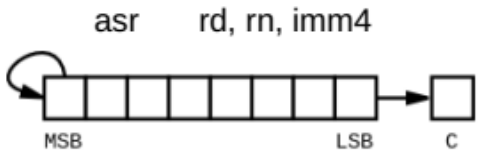
A *flag Negative (N)*, após uma operação aritmética ou lógica fica com valor lógico do *bit* de maior peso do resultado.

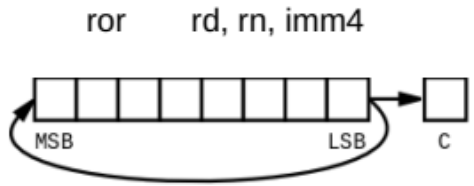
O resultado destas instruções são um registo qualquer do *register file.*
As instruções de processamento obedecem ao seguinte formato:

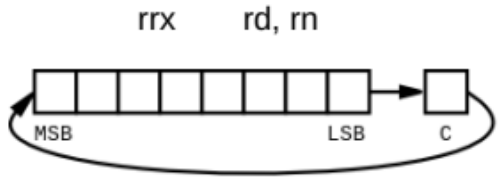












RD - registo destino/fonte (R0 a R15)
RN - registo do operando A (R0 a R7)
RM - registo do operando B (R0 a R7)
imm4 - constante de 4 bits sem sinal.

add	rd, rn, rm	1	0	0	0	0	rm	rn	rd
sub	rd, rn, rm	1	0	0	0	1	rm	rn	rd
adc	rd, rn, rm	1	0	0	1	0	rm	rn	rd
sbc	rd, rn, rm	1	0	0	1	1	rm	rn	rd
add	rd, rn, <imm4>	1	0	1	0	0	imm4	rn	rd
sub	rd, rn, <imm4>	1	0	1	0	1	imm4	rn	rd

cmp	rn, rm	1	0	1	1	1	rm	rn				

and	rd, rn, rm	1	1	0	0	0	rm	rn	rd
orr	rd, rn, rm	1	1	0	0	1	rm	rn	rd
eor	rd, rn, rm	1	1	0	1	0	rm	rn	rd
rrx	rd, rn	1	1	0	1	1		rn	rd

lsl	rd, rn, <imm4>	1	1	1	0	0	imm4	rn	rd
lsr	rd, rn, <imm4>	1	1	1	0	1	imm4	rn	rd
asr	rd, rn, <imm4>	1	1	1	1	0	imm4	rn	rd
ror	rd, rn, <imm4>	1	1	1	1	1	imm4	rn	rd

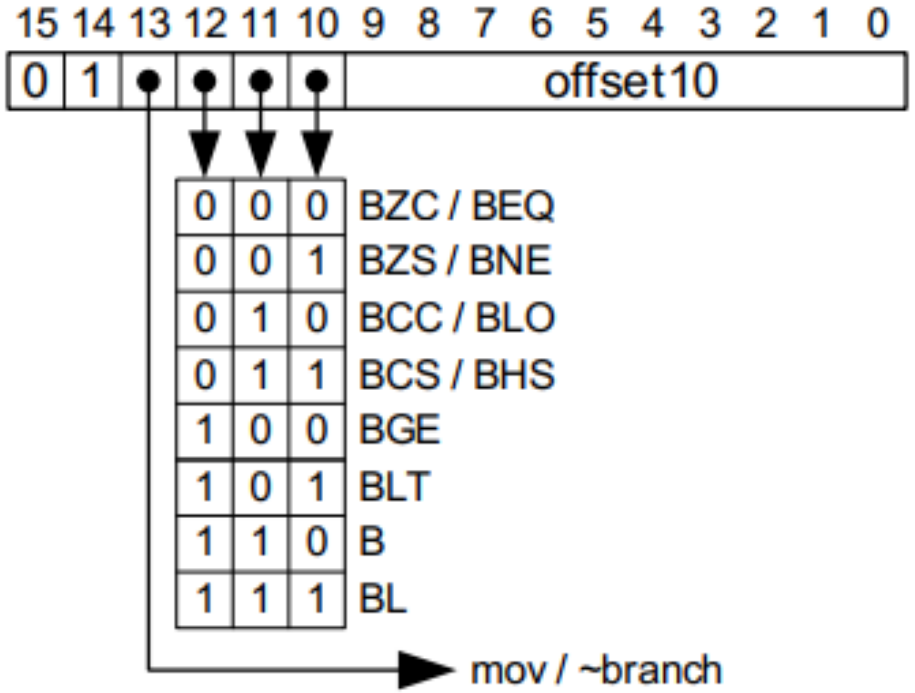
As instruções de controlo de fluxo condicional, fazem uso das *flags*.

Flag Zero (Z).
Flag Carry (C).
Flag Overflow (O).
Flag Negative (N).

- Nas instruções de salto condicional ou incondicional, o endereço é calculado pela soma do registo PC com um deslocamento especificado em código dos complementos para dois com 10 bits – **offset10**. A constante **offset10** é multiplicada por 2, isto porque **offset10** corresponde ao número de instruções a saltar. A constante **offset10 é** tomada como um valor inteiro com sinal, permitindo desta forma que o salto se realize para a frente ou para trás relativamente ao conteúdo do registo **PC**, ou seja, +510 ou menos -512 instruções. No cálculo de **offset10**, é necessário tomar em consideração que o valor do registo **PC** no momento da execução corresponde ao endereço imediatamente a seguir ao da instrução *branch,* pois o P16 realiza pré-preparação do **PC**.

- Para dar suporte à implementação de rotinas, o P16 põe disponível uma instrução específica – *branch with link* (**bl**) – que utiliza o registo R14 como registo de ligação (endereço de retorno).

As instruções de controlo de fluxo de execução obedecem ao seguinte formato:



beq/bzs	label	0	1	0	0	0	0	imm10
bne/bzc	label	0	1	0	0	0	1	imm10
blo/bcs	label	0	1	0	0	1	0	imm10
bhs/bcc	label	0	1	0	0	1	1	imm10
bge	label	0	1	0	1	0	0	imm10
blt	label	0	1	0	1	0	1	imm10
b	label	0	1	0	1	1	0	imm10
bl	label	0	1	0	1	1	1	imm10

ldr	rd, [pc, <imm6>]	0	0	0	0	1	1	imm6		rd
pop	rd	0	0	0	0	0	1			rd
push	rs	0	0	1	0	0	1			rs

ldr	rd, [rn, <imm3>]	0	0	0	0	0	0	imm3	rn	rd
ldrb	rd, [rn, <imm3>]	0	0	0	0	1	0	imm3	rn	rd
ldr	rd, [rn, rm]	0	0	0	1	0	rm		rn	rd
ldrb	rd, [rn, rm]	0	0	0	1	1	rm		rn	rd

str	rs, [rn, <imm3>]	0	0	1	0	0	0	imm3	rn	rs
strb	rs, [rn, <imm3>]	0	0	1	0	1	0	imm3	rn	rs
str	rs, [rn, rm]	0	0	1	1	0	rm		rn	rs
strb	rs, [rn, rm]	0	0	1	1	1	rm		rn	rs

add	rd, rn, rm	1	0	0	0	0	rm	rn	rd
sub	rd, rn, rm	1	0	0	0	1	rm	rn	rd
adc	rd, rn, rm	1	0	0	1	0	rm	rn	rd
sbc	rd, rn, rm	1	0	0	1	1	rm	rn	rd
add	rd, rn, <imm4>	1	0	1	0	0	imm4	rn	rd
sub	rd, rn, <imm4>	1	0	1	0	1	imm4	rn	rd

cmp	rn, rm	1	0	1	1	1	rm	rn	

and	rd, rn, rm	1	1	0	0	0	rm	rn	rd
orr	rd, rn, rm	1	1	0	0	1	rm	rn	rd
eor	rd, rn, rm	1	1	0	1	0	rm	rn	rd
rrx	rd, rn	1	1	0	1	1		rn	rd

lsl	rd, rn, <imm4>	1	1	1	0	0	imm4	rn	rd
lsr	rd, rn, <imm4>	1	1	1	0	1	imm4	rn	rd
asr	rd, rn, <imm4>	1	1	1	1	0	imm4	rn	rd
ror	rd, rn, <imm4>	1	1	1	1	1	imm4	rn	rd

mov	rd, rs	1	0	1	1	0	rs		0	0	0	rd
mvn/not	rd, rs	1	0	1	1	0	rs		0	0	1	rd
movs	pc, lr	1	0	1	1	0			0	1	0	
msr	cpsr, rs	1	0	1	1	0	rs		1	0	0	
msr	spsr, rs	1	0	1	1	0	rs		1	0	1	
mrs	rd, cpsr	1	0	1	1	0			1	1	0	rd
mrs	rd, spsr	1	0	1	1	0			1	1	1	rd

mov	rd, <imm8>	0	1	1	0	imm8		rd
movt	rd, <imm8>	0	1	1	1	imm8		rd

beq/bzs	label	0	1	0	0	0	0	imm10
bne/bzc	label	0	1	0	0	0	1	imm10
blo/bcs	label	0	1	0	0	1	0	imm10
bhs/bcc	label	0	1	0	0	1	1	imm10
bge	label	0	1	0	1	0	0	imm10
blt	label	0	1	0	1	0	1	imm10
b	label	0	1	0	1	1	0	imm10
bl	label	0	1	0	1	1	1	imm10