Programski model procesora ARM

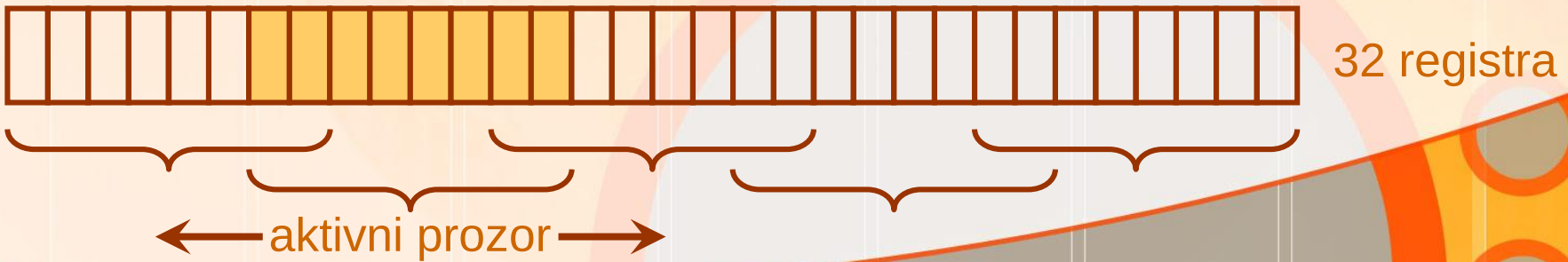
Kako bi objasnili arhitekturu procesora Arm izabrali smo jedan relativno jednostavniji processor iz Arm familije te će sve objašnjeno biti vezano za tu inačicu procesora:

Arm7

Osnovne karakteristike arhitekture procesora ARM

- Vrsta arhitekture
  - Procesor ARM po većini svojih karakteristika spada u grupu RISC procesora, ali ima i neke karakteristike CISC-a.
- Skup registara
  - relativno veliki, uniformni skup registara
  - nema općenitih registarskih prozora*

i * primjer za 5 prozora sa po 8 registara i s preklapanjem od 2 registra:



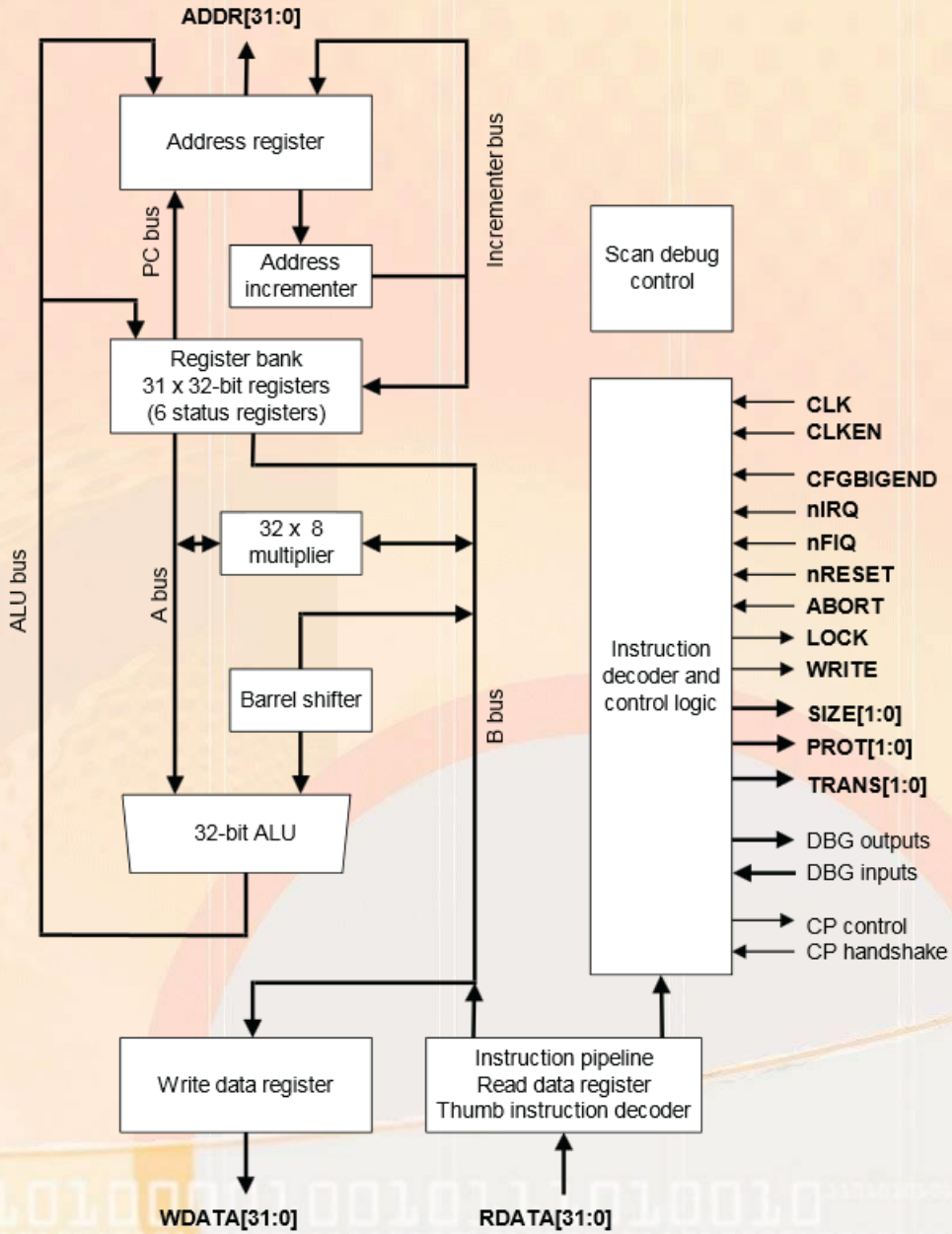
Skup naredaba

- "Load-store" arhitektura
  - Prednosti/nedostaci kao što smo ranije objašnjavali
- Sve naredbe su iste duljine od 32 bita
  - Prednosti/nedostaci kao što smo ranije objašnjavali
- Naredbe za obradu podataka sa 3 adrese (3 operanda)
- Uvjetno izvođenje gotovo svake naredbe
  - Drugi procesori uglavnom nemaju ovu mogućnost
  - Služi za ubrzanje malih odsječaka programskog koda
- Izvođenje općenitog pomaka i aritmetičko-logičke naredbe u jednom vremenskom periodu
  - Barrel shifter na ulazu u ALU
- "Thumb" skup naredaba
  - Za sustave kojima je važna minimizacija memorije

Opće karakteristike

- Mogućnost proširenja skupa naredaba i registara korištenjem koprocesorskih naredaba
  - Neki kodovi nisu iskorišteni te ih možemo sami definirati
- Protočna struktura
  - Protočna struktura od tri (ARM7), pet (ARM9), šest (ARM10) ili osam (ARM11) razina
  - Visoka efikasnost
- Memorijska arhitektura
  - Von Neumannova ili Harvardska arhitektura (različito za pojedine jezgre)
- Modularna arhitektura
  - Namijenjena za SoC (System on Chip)
- Mala potrošnja

ARM 7



Programski model procesora ARM

- Programski model - skup značajki određene arhitekture koji je na raspolaganju programeru:
  - tipovi podataka
  - procesorski načini rada
  - registri
    - registri opće namjene
    - registri programskih stanja
  - iznimke
  - memorija

Tipovi podataka

- Procesor ARM je 32-bitni procesor, ali može obrađivati i podatke manje širine. Programeru su na raspolaganju sljedeći tipovi podataka:
  - bajt (byte, 8 bitova), označava se slovom **B**
  - poluriječ (halfword, 16 bitova), označava se slovom **H**
  - riječ (word, 32 bita), označava se slovom **W**
- Iako ARM može koristiti kraće tipove podataka, sve osnovne operacije obavljaju se nad 32-bitnim riječima.

Procesorski načini rada

- ARM arhitektura podržava sedam procesorskih **načina rada** (engl. modes).
- Postoji jedan korisnički način rada (User) i šest privilegiranih načina rada
- U određenom procesorskom načinu omogućen je ili onemogućen pristup određenim resursima sustava
- Onemogućavanje pristupa nekim resursima u korisničkom načinu je od velike važnosti za izvedbu operacijskih sustava

- Procesorski način se definira postavljanjem najnižih 5 bitova u posebnom registru CPSR što će kasnije biti objašnjeno

Procesorski načini rada

Način	Oznaka	Opis načina
User	usr	Normalno izvođenje programa
System	sys	Podrška za izvođenje privilegiranih zadataka operacijskog sustava
Supervisor	svc	Zaštićen način za operacijski sustav
Abort	abt	Podrška za izvedbu virtualne memorije i zaštitu memorije
Undefined	und	Podrška za programsku emulaciju koprocesora
Interrupt	irq	Podrška za obradu općenitog prekida
Fast Interrupt	fiq	Podrška za brzi prekid

Privilegirani načini rada

"User" način

- Normalan način u kojem se izvode korisnički programi.
- U ovom načinu program ne može koristiti zaštićene resurse sustava te ne može promijeniti način rada.
- Pokušaj pisanja u CPSR[23:0] u User načinu rada procesor ignorira, tako da programi koji se izvode u User načinu ne mogu promijeniti način u neki od privilegiranih

Ostali procesorski načini

- Ostali načini smatraju se privilegiranim i služe pri izvođenju specifičnih operacija. Programi koji se izvode u privilegiranim načinima imaju potpuni pristup svim resursima sustava.
- Načini fiq, irq, svc, abt i und aktiviraju se kad se u sustavu generira iznimka
- Način sys namijenjen je izvođenju privilegiranih sistemskih funkcija, ali bez potrebe korištenja alternativnih registara. Obrada nekih važnijih iznimaka bit će objašnjena kasnije.
- Programi pisani u ATLASU imat će pristup svim resursima sustava.

Registri

- Procesor ARM ima ukupno 37 registara širine **32 bita**. Od ukupnog broja registara, postoji:
  - 31 registar opće namjene (uključujući i programsko brojilo).
    - Važno je uočiti da **u jednom trenutku možemo koristiti samo 16 registara** opće namjene (R0-R15) dok ostali nisu u to vrijeme dostupni !!!
  - 6 registara programskog stanja. Registri programskog stanja su također 32-bitni, no samo neki bitovi se koriste te ostali bitovi ne moraju nužno biti izvedeni u sklopovlju.

Registri

User	System	Superv.	Abort	Undefined	Interrupt	Fast Int.
R0						
R1						
R2						
R3						
R4						
R5						
R6						
R7						
R8						R8_fiq
R9						R9_fiq
R10						R10_fiq
R11						R11_fiq
R12						R12_fiq
R13		R13_svc	R13_abt	R13_und	R13_irq	R13_fiq
R14		R14_svc	R14_abt	R14_und	R14_irq	R14_fiq
R15=PC						
CPSR						
		SPSR_svc	SPSR_abt	SPSR_und	SPSR_irq	SPSR_fiq

Registri

- Registri opće namjene (R0-R15)
- Registri opće namjene, R0-R15 mogu se podijeliti u tri grupe:
  - Jednoznačno definirani registri R0-R7
  - Višeznačno definirani registri R8-R14
  - Programsko brojilo R15
- Registri programskog stanja (CPSR, SPSR)
- **NAPOMENA: u ATLAS-u su svi registri jednoznačni i procesor ima samo jedan način rada - System!!!**

vidi sliku na sljedećem slajdu >>>

Registri

User	System	Superv.	Abort	Undef.	Interrupt	Fast Int.
R0						
R1						
R2						
R3						
R4						
R5						
R6						
R7						
R8						R8_fiq
R9						R9_fiq
R10						R10_fiq
R11						R11_fiq
R12						R12_fiq
R13		R13_svc	R13_abt	R13_und	R13_irq	R13_fiq
R14		R14_svc	R14_abt	R14_und	R14_irq	R14_fiq
R15=PC						
CPSR						
		SPSR_svc	SPSR_abt	SPSR_und	SPSR_irq	SPSR_fiq

Jednoznačno definirani registri R0-R7

Višeznačno definirani registri R8-R14

→ Programsko brojilo

Registri prog. stanja

Jednoznačno definirani registri R0-R7

- Procesor ARM sadrži osam fizičkih registara nazvanih R0-R7.
- Registri R0-R7 su jednoznačno definirani, te će program pristupati istim fizičkim registrima bez obzira na to u kojem se načinu rada procesor nalazi.
- Ovi registri su, u punom značenju pojma, registri opće namjene jer za njih nije predviđeno nikakvo posebno značenje te se slobodno mogu koristiti u svim mogućim situacijama.
- Prema tome, za sve načine rada, registri R0- R7 su identični.

Višeznačno definirani registri R8-R14

- Za razliku od R0-R7, prilikom pristupa nekom od registara R8-R14 adresirat će se određeni fizički registar ovisno o tome u kojem načinu rada se procesor nalazi.

- Na primjer, ako se procesor nalazi u privilegiranom načinu Supervisor:
  - pri adresiranju registra R13 procesor će pristupiti posebnom fizičkom registru R13_svc koji je dostupan samo u ovom procesorskom stanju. Isto vrijedi i za registar R14.
  - prilikom pristupanja registrima koji nisu višeznačno definirani (R0-R12 i R15 za način Supervisor), uvijek se pristupa zajedničkim fizičkim registrima.

Višeznačno definirani registri R8-R14

- Razlog za korištenje dodatnih fizičkih registara R13 i R14 je u tome što se registri R13 i R14 koriste za posebne namjene.
- Registar R13 se (po dogovoru) koristi kao pokazivač na vrh stoga (SP, Stack Pointer) te je ovime omogućeno da se u svakom načinu rada može definirati neovisan stog.
- Registar R14 se koristi kao registar za pohranjivanje povratne adrese za vraćanje iz potprograma ili iznimke.
- Oba ova registra se mogu koristiti i kao registri opće namjene ukoliko ih sustav ne koristi za ove posebne namjene.

Višeznačno definirani registri R8-R14

- U načinu brzog prekida (fiq) adresira se skup od 7 posebnih fizičkih registara (R8_fiq-R14_fiq).

- To omogućuje da se obrada brzog prekida izvede čim brže - bez potrebe za spremanjem konteksta.

Programsko brojilo R15

- Registar R15 je **programsko brojilo** te se u većini slučajeva korištenje ovog registra za opće namjene ne preporuča, a često i nije dozvoljeno.
- Čitanjem podatka zapisanog u R15 dobije se vrijednost adrese trenutne naredbe + 8 bajtova (zbog protočne strukture).
- Valja uočiti da su pri dohvatu naredbe najniža dva bita uvijek postavljena u logičku nulu zbog toga jer su naredbe ARM-a uvijek poravnate na širinu riječi.
- Upis vrijednosti u R15 izvodi neku vrstu forsiranog skoka, no ovaj način korištenja se ne preporučuje.

Registri programskog stanja (CPSR, SPSR)

- U registru trenutnog programskog stanja CPSR (Current Program Status Register) spremljeni su bitovi koji definiraju različita stanja procesora i programa.

- Registar CPSR je dostupan u svim procesorskim načinima.

- Registar pohranjenog programskog stanja SPSR (Saved Program Status Register) koristi se kad procesor obrađuje iznimke. U njega se pohranjuje vrijednost registra CPSR

Bitovi registara CPSR/SPSR

31	30	29	28	27	....	7	6	5	4	3	2	1	0
N	Z	C	V	Q		I	F	T	M				

- Bitovi M4-M0: način rada procesora (npr, M[4:0]=10000 znači način "usr").
- Bit T označava da ARM izvodi Thumb naredbu.
- Postavljanje bita I ili F u 1 onemogućit će prekid (I) ili brzi prekid (F).
- Najviših pet bitova u CPSR predstavljaju zastavice:
  - N (Negative) - negativna vrijednost,
  - Z (Zero) - nula,
  - C (Carry) - prijenos,
  - V (oVerflow) - preljev.
  - Q je zastavica koja označava da je došlo do preljeva i-ili zasićenja kod proširenih DSP naredaba (ARM procesori u E izvedbi).
- Ostali bitovi ne koriste se i sačuvani su za buduća proširenja.

Iznimke

- Tijekom rada procesor često treba obraditi neke događaje, poput obrade prekida ili obrade nepostojeće naredbe. U takvim situacijama sustav će generirati **iznimke**.

- Iznimke se općenito obrađuju ovako:
  - procesor pohranjuje povratnu adresu u registar R14
  - procesor pohranjuje trenutno programsko stanje (CPSR) u registar SPSR
  - procesor se prebacuje u potrebno privilegirano stanje
  - procesor započinje s izvođenjem potprograma za obradu iznimke

- Više o iznimkama u posebnom poglavlju...

Zapis podataka u memoriji

- ARM arhitektura koristi jedinstveni memorijski adresni prostor od 2^{32} 8-bitnih podataka

- Podaci u memoriji organizirani su kao:
  - 8-bitni bajtovi (byte)
  - 16-bitne poluriječi (half-word)
  - 32-bitne riječi (word).

Zapis podataka u memoriji

- Pri normalnom pristupu, pretpostavlja se da su podaci poravnati na sljedeći način:
  - riječi (32 bita) su poravnate tako da počinju na adresi djeljivoj s 4 (npr. 0, 4, 8, ...)
  - poluriječi (16 bitova) su poravnate tako da počinju na parnoj adresi (npr. 0, 2, 4, 6, 8, ...)
  - bajtovi (8 bitova) su bilo gdje u memorijskom prostoru

Zapis podataka u memoriji

- Ako se podatak sastoji od nekoliko bajtova, tada je potrebno definirati kojim redosljedom se ti bajtovi zapisuju u memoriju (endianness).

- U inicijalnom stanju, procesor ARM podržava NV (Niži pa Viši) zapis bajtova tako da se na nižu memorijsku adresu zapisuje bajt manje važnosti (little-endian)*

* Kod procesora ARM moguće je odabrati i da zapisuje bajtove u redoslijedu VN (Viši pa Niži), tj. u redoslijedu big-endian

Pregled skupa naredaba procesora ARM

Skup naredaba procesora ARM - ukratko

- Sve naredbe iz skupa naredaba procesora ARM mogu se podijeliti u šest osnovnih grupa:
  - Naredbe load i store
  - Naredbe za obradu podataka
  - Naredbe grananja
  - Naredbe za prijenos registara stanja
  - Koprocesorske naredbe
  - Naredbe za generiranje iznimke

Naredbe load i store

- Programer ima na raspolaganju naredbe za prijenos podataka iz memorije u registar (load register, LDR) i iz registra u memoriju (store register, STR) u nekoliko različitih verzija.

- Najčešće korištene naredbe obavljaju prijenos samo jednog podatka koji može biti širok 8, 16 ili 32 bita
  - Može se odabrati predznačno proširivanje bitova do pune 32-bitne riječi (pri prijenosu bajta ili poluriječi) ili proširivanje ništicama
  - Postoji velik broj načina adresiranja

Naredbe load/store multiple, swap

- Postoje i naredbe za prijenos bloka podataka gdje se podaci spremaju ili uzimaju iz **više registara** (load/store multiple registers)
  - postoje razne modifikacije koje, na primjer, određuju kako su podaci smješteni u memoriji (povećanje/smanjenje pokazivača na podatak prije/poslije izvođenja naredbe).
  - efikasan način zapisivanja ili čitanja podataka sa stoga.

- U ovu grupu naredbi spadaju i naredbe za zamjenu sadržaja memorije i registara (swap) koje se najčešće koriste pri izvedbi semafora kod operacijskih sustava.

Naredbe za obradu podataka

- Naredbe za obradu podataka obavljaju razne operacije nad podacima koji se nalaze u registrima.

- Naredbe za obradu podataka dijele se na:
  - aritmetičko-logičke naredbe
  - naredbe za množenje i
  - naredba za brojenje vodećih nula.

Aritmetičko-logičke naredbe

- Imaju zajedničku karakteristiku da mogu obraditi do dva operanda i spremiti rezultat u zadani registar te, ako je zadano, osvježiti zastavice stanja.
- Od dva operanda koji se mogu koristiti u operaciji jedan uvijek mora biti registar, a drugi može biti ili neposredna vrijednost ili vrijednost registra nad kojom se još može obaviti određen pomak.
- U okviru aritmetičko-logičkih naredbi postoje i četiri naredbe za usporedbu koje su definirane vrlo slično aritmetičko-logičkim naredbama s razlikom da se rezultat nigdje ne sprema i da se uvijek osvježavaju zastavice stanja.

Naredbe za množenje

- Množenje se uvijek obavlja nad dva 32-bitna podatka.
- Različite naredbe za množenje omogućuju da se spremaju samo 32 bita rezultata u izabrani registar ili da se spreme svih 64 bita rezultata u dva registra.
- U oba slučaja moguće je još omogućiti zbrajanje sa prethodnom vrijednosti rezultantnih registara, čime se ostvaruje tzv. operacija MAC (Multiply-And-Accumulate) koja se vrlo često koristi u algoritmima za obradu signala.

Naredba clz

- Naredba CLZ (Count Leading Zeroes) služi za brojenje vodećih nula u podatku.

- Ovo je specifična naredba koja je veoma korisna kod izvedbe matematičkih algoritama (normiranje brojeva) i algoritama za kompresiju (npr. metode duljine niza).

Naredbe grananja

- Programsko brojilo kod procesora ARM je izvedeno kao običan registar te je grananja moguće izvesti bilo kojom naredbom load ili nekom aritmetičko/logičkom naredbom.
- Pored toga ARM definira i standardne naredbe za grananje poput:
  - Branch (B)
  - Branch and Link (BL).
- Posebnost procesora ARM je što neke izvedbe posjeduju takozvani 'Thumb' skup naredaba te postoje specijalne naredbe grananja BX kojima programer prebacuje procesor iz stanja dekodiranja standardnog skupa naredaba u 'Thumb' skup naredaba i obratno.

Naredbe za pristup registrima stanja

- Posebna grupa naredaba omogućuje prijenos podataka iz registara stanja (CPSR, SPSR) procesora ARM u neki registar opće namjene i obratno.

- Pisanjem u CPSR, na primjer, programer može postaviti stanja zastavica, stanja bitova za omogućavanje prekida kao i procesorska stanja.

Koprocesorske naredbe

- Služe za komunikaciju procesora ARM i koprocesora ako se oni nalaze u sustavu (prijenos podataka, operacije nad podacima, ...)

Naredbe za generiranje iznimke

- Omogućuju programeru i sistemu da se pokrene proces obrade iznimke

- Programski ekvivalent sklopovskim iznimkama koje se postavljaju na priključke procesora

Strojni kodovi ARM-a

- Procesor ARM ima strojne kodove fiksne širine 32 bita (osim za naredbe tzv. "Thumb" skupa)

- Ova fiksna duljina koda naredbe ima svoje prednosti i nedostatke.
  - Prednost je jednostavnost izvođenja i efikasnost protočne strukture za naredbe
  - Nedostatak je ograničenost broja bitova za opis pojedinih dijelova naredbe.

- Formati strojnih kodova dosta ovise o pojedinoj naredbi (vidi primjere na sljedećem slajdu)

Primjeri formata naredaba

Polje uvjeta

Polje uvjeta (condition field)

- Većina ARM-ovih naredaba **može se izvoditi uvjetno**. Te naredbe u strojnom kodu imaju **polje uvjeta** (condition field) u kojem se zadaje uvjet koji mora biti zadovoljen da bi se naredba izvela.
- Svi uvjeti temelje se na zastavicama stanja u CPSR.
- Ako uvjet nije zadovoljen, umjesto naredbe izvest će se NOP*.
- Uvjetno izvođenje omogućuje programeru izvedbu programa bez korištenja naredbi grananja, čime se može ubrzati izvođenje nekog kratkog dijela programa

* NOP je naredba koja ne izvodi ništa (kratica od No Operation)

Uvjeti

Opcode [31:28]	Mnemonički naziv	Puni naziv (engleski)	Stanje zastavica
0000	EQ	Equal	Z
0001	NE	Not equal	!Z
0010	CS/HS	Carry set/unsigned higher or same	C
0011	CC/LO	Carry clear/unsigned lower	!C
0100	MI	Minus/negative	N
0101	PL	Plus/positive or zero	!N
0110	VS	Overflow	V
0111	VC	No overflow	!V
1000	HI	Unsigned higher	C and !Z
1001	LS	Unsigned lower or same	!C or Z
1010	GE	Signed greater than or equal	N == V
1011	LT	Signed less than	N != V
1100	GT	Signed greater than	!Z and N == V
1101	LE	Signed less than or equal	Z or N != V
1110	AL	Always (unconditional)	-
1111	(NV)	See Condition code 0b1111	-