spaces		**examples**
` `	🟩	
`         `	🟩	

comment		**examples**
`// comment`	🟩	
`// comment`	🟩	
`//`	🟩	
`//`	🟩	

comma		**examples**
`,`	🟩	
`    ,`	🟩	

linebreak		**examples**
` `	🟩	

`// comment`
 	
`// eh`
	🟩	

linebreak		**counterexamples**

`       x`
	🟩	**Parsing error:** The parser did not consume the entire input.
		`1 | ¤`
		`   ~|` Surplus characters on next line
		`    |` Valid `<linebreak>`
		Hint: halted due to the following:
		`2 | ␣␣␣␣␣␣␣x`
		`             ^` Can't match string "
		"
		While trying to parse: `<linebreak>` → `<lb>`.

`// comment`
`   x`
	🟩	**Parsing error:** The parser did not consume the entire input.
		`1 | //␣comment¤`
		`   ~~~~~~~~~~~|` Surplus characters on next line
		`              |` Valid `<linebreak>`
		Hint: halted due to the following:
		`2 | ␣␣␣x`
		`         ^` Can't match string "
		"
		While trying to parse: `<linebreak>` → `<lb>`.

linebreak		**counterexamples**

`// comment`
 	
`   x`
	🟩	**Parsing error:** The parser did not consume the entire input.
		`2 | ¤`
		`   ~|` Surplus characters on next line
		`    |` Valid `<linebreak>`
		Hint: halted due to the following:
		`3 | ␣␣␣x`
		`         ^` Can't match string "
		"
		While trying to parse: `<linebreak>` → `<lb>`.

`// comment`
`   x`
	🟩	**Parsing error:** The parser did not consume the entire input.
		`1 | //␣comment¤`
		`    ~~~~~~~~~~~~|` Surplus characters on next line
		`                |` Valid `<linebreak>`
		Hint: halted due to the following:
		`2 | ␣␣␣x`
		`         ^` Can't match string "
		"
		While trying to parse: `<linebreak>` → `<lb>`.

ident-startchar		**examples**
`f`	🟩	f
`A`	🟩	A

ident-startchar		**counterexamples**
`_`	🟩	**Parsing error:** The input does not match the expected format. `1 | _` `    ^` Character is out of range While trying to parse: `<ident-startchar>`.

ident-anychar		**examples**
`A`	🟩	A
`0`	🟩	0
`x`	🟩	x
`_`	🟩	_

ident		**examples**
`LD_foo`	🟩	`(lab: "LD_foo")`
`main`	🟩	`(lab: "main")`

ident		examples
`loop0`	🟩	`(lab: "loop0")`

ident		counterexamples

`0var`	🟩	**Parsing error:** The input does not match the expected format.
		`1 | 0var`
		`^ Character is out of range`
		While trying to parse: `<ident>` → `<ident-startchar>`.

size		examples
`.byte`	🟩	`(size: 1)`
`.word`	🟩	`(size: 4)`
`.hword`	🟩	`(size: 2)`

size		counterexamples
`.bte`	🟩	**Parsing error:** The input does not match the expected format. `1 | .bte` ^ Can't match string "`word`" While trying to parse: `<size>`.

hexvalue		examples
`0x45`	🟩	`(hex: "45")`
`0xdead`	🟩	`(hex: "dead")`
`0xffffffff`	🟩	`(hex: "ffffffff")`

hexvalue		counterexamples

`0xz`	🟩	**Parsing error:** The input does not match the expected format.
		`1 | 0xz`
		`^ Character is out of range`
		While trying to parse: `<hexvalue>`.

decvalue		examples
`42`	🟩	`(int: "42")`
`1000`	🟩	`(int: "1000")`
`0`	🟩	`(int: "0")`

decvalue		counterexamples

`45x`	🟩	**Parsing error:** The parser did not consume the entire input.
		`1 | 45x`
		`     ~~|^ Surplus characters`
		`       | Valid <decvalue>`
		Hint: halted due to the following:
		`1 | 45x`
		`        ^ Character is out of range`
		While trying to parse: `<decvalue>`.

value		examples
`0x44`	🟩	`(hex: "44")`
`420`	🟩	`(int: "420")`
`mask`	🟩	`(lab: "mask")`

value		counterexamples

`0var`	🟩	**Parsing error:** The parser did not consume the entire input.
		`1 | 0var`
		`     ~|^^^ Surplus characters`
		`      | Valid <value>`
		Hint: halted due to the following:
		`1 | 0var`
		`       ^ Character is out of range`
		While trying to parse: `<value>` → `<decvalue>`.

concrete-value		examples
`.byte 0x42`	🟩	`(size: 1, hex: "42")`
`.word mask`	🟩	`(size: 4, lab: "mask")`

concrete-value		counterexamples

`.bte 0x42`	🟩	**Parsing error:** The input does not match the expected format.
		`1 | .bte␣0x42`
		`       ^ Can't match string "word"`
		While trying to parse: `<concrete-value>` → `<size>`.

concrete-value		**counterexamples**

`.byte 45x`	🟩	**Parsing error:** The parser did not consume the entire input.
		`1 | .byte␣45x`
		`    ~~~~~~~~|^ Surplus characters`
		`            | Valid <concrete-value>`
		Hint: halted due to the following:
		`1 | .byte␣45x`
		`             ^ Character is out of range`
		While trying to parse: `<concrete-value>` → `<value>` → `<decvalue>`.

abstract-value		**examples**
`.skip 5`	🟩	`(size: 5)`

abstract-value		**counterexamples**

`skip 3`	🟩	**Parsing error:** The input does not match the expected format.
		`1 | skip␣3`
		`    ^ Can’t match string “.skip”`
		While trying to parse: `<abstract-value>`.

`.skip 0x42`	🟩	**Parsing error:** The parser did not consume the entire input.
		`1 | .skip␣0x42`
		`    ~~~~~~~|^^^ Surplus characters`
		`           | Valid <abstract-value>`
		Hint: halted due to the following:
		`1 | .skip␣0x42`
		`            ^ Character is out of range`
		While trying to parse: `<abstract-value>` → `<decvalue>`.

data-value		**examples**
`.byte 0x42`	🟩	`(size: 1, hex: "42")`
`.skip 8`	🟩	`(size: 8)`

data-label		**examples**
`a:`	🟩	`(lab: "a")`
`B:`	🟩	`(lab: "B")`

data-contents		**examples**

`A:`
`  B:`
`  .word 0x64`
 	
`  .byte 0x42`
 	
`foo: .hword 42`
`bar: baz: .byte`
`1 .byte 2 .byte 3`
	🟩	`(`
		`  (lab: "A"),`
		`  (lab: "B"),`
		`  (size: 4, hex: "64"),`
		`  (size: 1, hex: "42"),`
		`  (lab: "foo"),`
		`  (size: 2, int: "42"),`
		`  (lab: "bar"),`
		`  (lab: "baz"),`
		`  (size: 1, int: "1"),`
		`  (size: 1, int: "2"),`
		`  (size: 1, int: "3"),`
		`)`

data-contents		**counterexamples**

`.byte 1 c`	🟩	**Parsing error:** The parser did not consume the entire input.
		`1 | .byte␣1␣c`
		`    ~~~~~~~~~|^ Surplus characters`
		`             | Valid <data-contents>`
		Hint: halted due to the following:
		`1 | .byte␣1␣c`
		`              ^ Can’t match string “:”`
		While trying to parse: `<data-contents>` → `<data-label>`.

data-section		**examples**

`  .data`
`foo: .word 0x42`
`bar: .word 0x43`
	🟩	`(`
		`  data: (`
		`    (lab: "foo"),`
		`    (size: 4, hex: "42"),`
		`    (lab: "bar"),`
		`    (size: 4, hex: "43"),`
		`  ),`
		`)`

instr-code		**examples**
`ldr`	🟩	`(instr: "ldr", size: 4)`
`add`	🟩	`(instr: "add")`
`lsl`	🟩	`(instr: "lsl")`
`ldrh`	🟩	`(instr: "ldr", size: 2)`
`b`	🟩	`(instr: "b")`

register-number		**examples**
`r0`	🟩	`(reg: 0)`

register-number		examples
`r9`	🟩	`(reg: 9)`
`r12`	🟩	`(reg: 12)`

register-alias		examples
`sp`	🟩	`(reg: 13)`
`lr`	🟩	`(reg: 14)`
`pc`	🟩	`(reg: 15)`

register		examples
`r8`	🟩	`(reg: 8)`
`sp`	🟩	`(reg: 13)`
`lr`	🟩	`(reg: 14)`
`r0`	🟩	`(reg: 0)`

constant		examples
`#4`	🟩	`(int: "4")`
`#0x1`	🟩	`(hex: "1")`

deref-offset		examples
`, r1`	🟩	`(reg: 1)`
`, #3`	🟩	`(int: "3")`

deref-reg		examples
`[r1]`	🟩	`(deref: ((reg: 1),))`
`[ r2  ]`	🟩	`(deref: ((reg: 2),))`
`[r1, r2]`	🟩	`(deref: ((reg: 1), (reg: 2)))`
`[r1 , #1]`	🟩	`(deref: ((reg: 1), (int: "1")))`
`[r1, r2, r3]`	🟩	`(deref: ((reg: 1), (reg: 2), (reg: 3)))`
`[r1, r2, #2]`	🟩	`(deref: ((reg: 1), (reg: 2), (int: "2")))`

local-label		examples
`.LD_foo`	🟩	`(lab: ".LD_foo")`

operand		examples
`lr`	🟩	`(reg: 14)`
`#1`	🟩	`(int: "1")`
`[ r1 , #2 ]`	🟩	`(deref: ((reg: 1), (int: "2")))`
`.LD_xx`	🟩	`(lab: ".LD_xx")`
`=mask`	🟩	`(eq: "mask")`
`loop0`	🟩	`(lab: "loop0")`

operands		examples
`lr, #1`	🟩	`((reg: 14), (int: "1"))`

`r0 , r0 , [r1, #2]`	🟩	`(`
		`  (reg: 0),`
		`  (reg: 0),`
		`  (deref: ((reg: 1), (int: "2"))),`
		`)`

instruction		examples

`ldr r0, [r1]`	🟩	`(`
		`  instr: "ldr",`
		`  size: 4,`
		`  ops: ((reg: 0), (deref: ((reg: 1),))),`
		`)`

`add r0, r1, r2`	🟩	`(instr: "add", ops: ((reg: 0), (reg: 1), (reg: 2)))`
`sub r1, #1`	🟩	`(instr: "sub", ops: ((reg: 1), (int: "1")))`
`lsl r1, #8`	🟩	`(instr: "lsl", ops: ((reg: 1), (int: "8")))`

inline-data		examples
`.LD_xx: .word x`	🟩	`(lab: "x", size: 4)`

`.LD_xx:`
`  .byte 0x42`
	🟩	`(lab: ".LD_xx", size: 1, hex: "42")`

inline-label		examples
`main:`	🟩	`(tag: "main")`

print-width		examples
`/8`	🟩	8
`/16`	🟩	16

register-slice		examples
`[:]`	🟩	`(start: auto, len: auto)`
`[1:]`	🟩	`(start: (int: "1"), len: auto)`
`[16:8]`	🟩	`(start: (int: "16"), len: (int: "8"))`

print-register		examples
`r0`	🟩	`(reg: 0)`
`r0[:8]`	🟩	`(reg: 0, start: auto, len: (int: "8"))`
`r0[16:32]`	🟩	`(reg: 0, start: (int: "16"), len: (int: "32"))`

print-list		examples
`r0, r1, r2`	🟩	`((reg: 0), (reg: 1), (reg: 2))`

print-directive		examples
`print/8 : r0`	🟩	`(print: (width: 8, regs: ((reg: 0),)))`
`print : r1, r2`	🟩	`(print: (width: auto, regs: ((reg: 1), (reg: 2))))`

directive		examples
`@ print/8: r0`	🟩	`(print: (width: 8, regs: ((reg: 0),)))`

text-contents		examples

`main:`
`  ldr r0, [r1]`
`  add r0, #1 // test`
`loop0:`
`  eor r0, r1, r2`
`.LD_data: .word data`
	🟩	`(`
		`  (tag: "main"),`
		`  (`
		`    instr: "ldr",`
		`    size: 4,`
		`    ops: ((reg: 0), (deref: ((reg: 1),))),`
		`  ),`
		`  (instr: "add", ops: ((reg: 0), (int: "1"))),`
		`  (tag: "loop0"),`
		`  (instr: "eor", ops: ((reg: 0), (reg: 1), (reg: 2))),`
		`  (lab: "data", size: 4),`
		`)`

text-section		examples

`  .text`
 	
`main:`
`  mov r0, #1 // test`
`  add r0, r0, r0`
	🟩	`(`
		`  text: (`
		`    (tag: "main"),`
		`    (instr: "mov", ops: ((reg: 0), (int: "1"))),`
		`    (instr: "add", ops: ((reg: 0), (reg: 0), (reg: 0))),`
		`  ),`
		`)`

arm		examples

.data
A: .byte 0x64
   .word 0x95
   .hword 45
 	
  .text
main: // main function
  ldr r0, .LD_A
  ldr r0, [r6]
  ldrb r0, [r6]
  ldrh r0, [r5, #1] // yay
  mov r5, r6
  mvn r5, r6
  mov r5, #0x1
  add r5, r6, #1
 	
  lsl r5, #1
  lsr r5, #1 // do some stuff
  // testing comments
  eor r3, r4, r5
  orr r3, r4, r5
  and r3, r4, r5
 	
.LD_A: .word A
	🟩	(
		  data: (
		    (lab: "A"),
		    (size: 1, hex: "64"),
		    (size: 4, hex: "95"),
		    (size: 2, int: "45"),
		  ),
		  text: (
		    (tag: "main"),
		    (
		      instr: "ldr",
		      size: 4,
		      ops: ((reg: 0), (lab: ".LD_A")),
		    ),
		    (
		      instr: "ldr",
		      size: 4,
		      ops: ((reg: 0), (deref: ((reg: 6),))),
		    ),
		    (
		      instr: "ldr",
		      size: 1,
		      ops: ((reg: 0), (deref: ((reg: 6),))),
		    ),
		    (
		      instr: "ldr",
		      size: 2,
		      ops: ((reg: 0), (deref: ((reg: 5), (int: "1")))),
		    ),
		    (instr: "mov", ops: ((reg: 5), (reg: 6))),
		    (instr: "mvn", ops: ((reg: 5), (reg: 6))),
		    (instr: "mov", ops: ((reg: 5), (hex: "1"))),
		    (
		      instr: "add",
		      ops: ((reg: 5), (reg: 6), (int: "1")),
		    ),
		    (instr: "lsl", ops: ((reg: 5), (int: "1"))),
		    (instr: "lsr", ops: ((reg: 5), (int: "1"))),
		    (instr: "eor", ops: ((reg: 3), (reg: 4), (reg: 5))),
		    (instr: "orr", ops: ((reg: 3), (reg: 4), (reg: 5))),
		    (instr: "and", ops: ((reg: 3), (reg: 4), (reg: 5))),
		    (lab: "A", size: 4),
		  ),
		)

total	111
🟩	111
🟥	0