Assembler Control Directives

Directive	Description	Syntax	Example
.arm	Following opcodes use the ARM instruction set	.arm	**.arm**
.thumb	Following opcodes use the THUMB instruction subset	.thumb	**.thumb**
.code 16	Same as **.thumb**	.code 16	**.code 16**
.code 32	Same as **.arm**	.code 32	**.code 32**
.include	Include a file.	.include "file"	**.include** "hardware.i"
.align	Byte align the following code to *alignment* byte boundary (**default=4**). Fill skipped bytes with *fill* (**default=0 or NOP**). If the number of bytes skipped is greater than max, then don't align (**default=*alignment***).	.align {alignment} {, fill} {, max}	**.align**
.balign	Same as **.align**	.balign {alignment} {, fill} {, max}	**.balign 8, 0**
.balignw	Half-word align the following code to *alignment* byte boundary (**default=4**). Fill skipped half-words with *fill* (**default=0 or NOP**). If the number of bytes skipped is greater than max, then don't align (**default=*alignment***).	.balignw {alignment} {, fill} {, max}	**.balignw 2**
.balignl	Word align the following code to *alignment* byte boundary (**default=4**). Fill skipped words with *fill* (**default=0 or NOP**). If the number of bytes skipped is greater than max, then don't align (**default=*alignment***).	.balignl {alignment} {, fill} {, max}	**.balignl**
.end	Marks the end of the assembly file. Data following this directive is not processed.	.end	**.end**
.fail	Generates errors or warnings durring assembly. If *expr* is greater than or equal to 500, it prints a warning message. If less than it prints an error message.	.fail expr	**.fail** 1
.err	Generate an error durring assembly.	.err	**.err**
.print	Print a string to standard output durring assembly.	.print string	**.print** "Something is broken"
.section	Tell the assembler to assemble the following in section *expr*. *expr* can be either **.text**, **.data**, or **.bss**.	.section expr	**.section** .bss
.text	Tell assembler to assemble the following in the "text" (code) section. You can also specify a subsection of "text" with *subsection*.	.text {subsection}	**.text**
.data	Tell assembler to assemble the following in the "data" section. You can also specify a subsection of "data" with *subsection*.	.data {subsection}	**.data 0**
.bss	Tell assembler to assemble the following in the "bss" (variables) section. You can also specify a subsection of "bss" with *subsection*.	.bss {subsection}	**.bss**
.struct	Tell assembler to assemble the following in an absolute section. Be sure to switch sections before you get back to code or data. feild1 is 0. field2 is 4.	.struct expr	**.struct** 0 field1: **.struct** field1 + 4 field2:
.org	Following code is inserted at the start of the section plus *new-lc*.	.org new-lc {, fill}	**.org** 0x20000
.pool	Tell the assembler where it can safely place data for immediate 32bit loads (ideally **after** your return). Use the = prefix operator to pool the value. **ldr r0, =0x4000002**	.pool	**.pool**

Symbol Directives

Directive	Description	Syntax	Example
.equ	Set the value of *symbol* equal to *expr*.	.equ symbol, expr	**.equ** Version, "0.1"
.set	Same as **.equ**	.set symbol, expr	**.set** Flavor, "CHERRY"
.equiv	Set the value of *symbol* equal to *expr*. Generates an error if the symbol has been previously defined.	.equiv symbol, expr	**.equiv** Version, "0.2"
.global	Makes *symbol* visible to the linker.	.global symbol	**.global** MyAsmFunc
.globl	Same as **.global**	.globl symbol	**.globl** MyOtherAsmFunc

Constant Definition Directives

Directive	Description	Syntax	Example
.byte	Define byte *expr* (8bit numbers)	.byte expr {, …}	**.byte** 25, 0x11, 031, 'A
.hword	Define half-word *expr* (16bit numbers)	.hword expr {, …}	**.hword** 2, 0xFFE0
.short	Same as **.hword**	.short expr {, …}	**.short** 257
.word	Define word *expr* (32bit numbers)	.word expr {, …}	**.word** 144511, 0x11223
.int	Same as **.word**	.int expr {, …}	**.int** 21
.long	Same as **.word**	.long expr {, …}	**.long** 1923, 0b10010101
.ascii	Define string *expr* (**non zero** terminated array of bytes)	.ascii expr {, …}	**.ascii** "Ascii text is here"
.asciz	Define string *expr* (**zero** terminated array of bytes)	.asciz expr {, …}	**.asciz** "Zero Terminated Text"
.string	Same as **.asciz**	.string expr {, …}	**.string** "My Cool String\n"
.quad	Define bignum *expr* (break at 8bit increments)	.quad expr {, …}	**.quad** 0xDAFADAFA911
.octa	Define bignum *expr* (break at 16bit increments)	.octa expr {, …}	**.octa** 0xFEDCBA987654321
.float	Define 32bit IEEE flonum *expr* (floating point numbers)	.float expr {, …}	**.float** 0f3.14, 0F359.2e11
.single	Same as **.float**	.single expr {, …}	**.single** 0f12341243.14E2
.double	Define 64bit IEEE flonum *expr* (floating point numbers)	.double expr {, …}	**.double** 0f2E1
.fill	Generate *repeat* copies of *value* that is of size *size*. *size* defaults to 1, and *value* defaults to 0.	.fill repeat {, size} {, value}	**.fill** 32, 4, 0xFFFFFFFF
.zero	Fills in *size* bytes with 0.	.zero size	**.zero** 400
.space	Fills in *size* bytes with *value*. *value* defaults to 0.	.space size {, value}	**.space** 25, 0b11001100
.skip	Same as **.space**	.skip size {, value}	**.skip 22**

Assembly Listing Directives

Directive	Description	Syntax	Example
.eject	Force a page break when generating assembly listings.	.eject	**.eject**
.psize	Set the number of *lines* to generate for each page of the assembly listing and the number of *columns*. Lines defaults to 60, Columns defaults to 200. A page break is generated when the number of lines hits *lines*. If *lines* is 0, then no page breaks are generated (excluding ones by **.eject**).	.psize lines {, columns}	**.psize** 40, 80
.list	Start generation of an assembly listings from **.list** to **.nolist**.	.list	**.list**
.nolist	End generation of an assembly listing. Listings can be re-started with **.list** again.	.nolist	**.nolist**
.title	Uses *heading* as the title (2nd line, under filename and page number)	.title "heading"	**.title** "My Asm Output"
.sbttl	Uses *heading* as the title (3rd line, under **.title**)	.sbttl "heading	**.sbttl** "Part 1: Cool stuff"

Conditional Directives

Directive	Description	Syntax	Example
.if	Assembles if *absolute_expression* does not equal zero. For all If's, if *absolute_expression* is omited, it equals 0.	.if {absolute_expression}	**.if (2+2)**
.elseif	Assembles if *absolute_expression* does not equal zero. Used in **.if** blocks to provide alternates when previous **.if**'s or **.elseif**'s fail.	.elseif {absolute_expression}	**.elseif (2+3) - 5**
.else	Assembles if all previous **.if** and **.elseif** blocks failed.	.else	**.else**
.endif	Ends an **.if** block.	.endif	**.endif**
.ifdef	Assembles if *symbol* exists.	.ifdef symbol	**.ifdef** _test_i_
.ifndef	Assembles if *symbol* does not exist.	.ifndef symbol	**.ifndef** _test_i_
.ifnotdef	Same as **.ifndef**	.ifnotdef symbol	**.ifnotdef** _test_i_
.ifc	Assembles if the strings are the same.	.ifc string1, string2	**.ifc** "this", "that"
.ifnc	Assembles if the strings are **not** the same.	.ifnc string1, string2	**.ifnc** "this", "that"
.ifeqs	Same as **.ifc**	.ifeqs string1, string2	**.ifeqs** "those", "this"
.ifnes	Same as **.ifeqs**	.ifnes string1, string2	**.ifnes** "those", "this"
.ifeq	Assembles if *absolute_expression* equals zero.	.ifeq {absolute_expression}	**.ifeq (2+2) - 4**
.ifne	Assembles if *absolute_expression* does not equal zero.	.ifne {absolute_expression}	**.ifne (2+2) - 5**
.ifge	Assembles if *absolute_expression* is greater than or equal to zero.	.ifge {absolute_expression}	**.ifge 10**
.ifgt	Assembles if *absolute_expression* is greater than zero.	.ifgt {absolute_expression}	**.ifgt**
.ifle	Assembles if *absolute_expression* is less than or equal to zero.	.ifle {absolute_expression}	**.ifle**
.iflt	Assembles if *absolute_expression* is less than zero.	.iflt {absolute_expression}	**.iflt -10**

Debug Directives

Directive	Description	Syntax	Example
.func	Generate debug information for code as a function. If *label* is omited, label is assumed to be *name*.	.func name {, label}	**.func** CoolFunc
.endfunc	Mark the end of a function.	.endfunc	**.endfunc**
.stabs	See GAS documentation for info. Not very useful unless your generate assembly listings from another source.	.stabs string, type, other, desc, value	

Looping Directives

Directive	Description	Syntax	Example
.rept	Repeat the sequence of lines between **.rept** and **.endr** *count* number of times.	.rept count	**.rept** 10
.irp	Evaluate a comma delimited sequence of statements to assign to the value of *symbol*.	.irp symbol, values…	**.irp** newval, 1, 2, 3
.irpc	For each character in *values*, assign its value.	.irp symbol, value	**.irp** newval, 123
	Symbol can be referenced with \symbol.	\symbol	.byte 0xC**\newval**
.endr	End **.rept**, **.irp**, and **.irpc** sequences.	.endr	**.endr**

Macro Directives

Directive	Description	Syntax	Example
.macro	Define a macro.	.macro name {args, …}	
	A macro can be defined without arguments,		**.macro** NoArgsMacro
	and can be called simply by specifying its name.	name {args, …}	**NoArgsMacro**
	A macro can also be defined with arguments,		**.macro** ArgMacro arg, arg2
	and can be called the same way with commas seperating its arguments.		**ArgMacro 10, 11**
	Arguments can be accessed by their name prefixed with a \.	\arg	mov r0, **\arg**
	You can define default macro arguments.		**.macro** ArgMacro arg=1, arg2
	Arguments are omited by simply placing a comma and no value, or ignoring them all together (trailing only).		**ArgMacro , 11**
	Arguments can be set in a modified order by referencing them by name.		**ArgMacro arg2=11, arg=10**
	Macros can be recursive.		
.endm	Mark the end of a macro.	.endm	**.endm**
.exitm	Exit a macro early.	.exitm	**.exitm**
\@	Psudo variable that contains the macro number executed. Can be used for a unique number on every macro definition.	\@	MyLabel**\@**:
.purgem	Undefine a macro, so that further uses do not evaluate.	.purgem name	**.purgem** NoArgsMacro

reEjected GNU AS ARM Reference V2

re-eject.gbadev.org

Digit Encoding Formats

Number Type	Base	Prefix	Digits	Example
Decimal Integer	10		**0 - 9**	25
Hexadecimal Integer	16	**0x** or **0X**	**0 - 9**, **A - F** (10 - 15)	0xD7
Octal Integer	8	**0**	**0 - 7**	027
Binary Integer	1	**0b** or **0B**	**0 - 1**	0b11010
Floating Point Number	10	**0f** or **0F**	**0 - 9**	0f+24.112E-25
Character	n/a	**'**	Ascii Symbol	'c
String	n/a	**"** and **"**	Ascii Symbol(s)	"MyString\n"
Number Type	**Base**	**Prefix**	**Digits**	**Example**

Escape Codes

\	Description	Ascii		\	Description
\b	Backspace	8		**\###**	Octal Character Code
\f	Form Feed	12		**\x##**	Hex Character Code
\n	New Line	10		****	\ character
\r	Carrige Return	13		**\"**	" character
\t	Horizontol Tab	9			

Expression Operators

Precedence	Symbol	Name	Operation
prefix	**-**	Negate	Negate argument.
prefix	**~**	Compliment	Compliment argument.
Highest	*****	Multiplication	Multiply arg1 by arg2.
Highest	**/**	Division	Divide arg1 by arg2.
Highest	**%**	Remainder	Divide arg1 by arg2, and return the remainder.
Highest	**<< or <**	Left Shift	Shift arg1 left by arg2.
Highest	**>> or >**	Right Shift	Shift arg1 right by arg2.
Intermediate	**|**	Bitwise OR	OR arg1 with arg2.
Intermediate	**&**	Bitwise AND	AND arg1 with arg2.
Intermediate	**^**	Bitwise XOR	XOR arg1 with arg2.
Intermediate	**!**	Bitwise OR NOT	OR arg1 with arg2, and NOT result.
Lowest	**+**	Addition	Add arg1 to arg2.
Lowest	**-**	Subtraction	Subtract arg2 from arg1.
Precedence	**Symbol**	**Name**	**Operation**