Next: ARM Opcodes, Previous: ARM Floating Point, Up: ARM-Dependent
[Contents][Index]

9.4.4 ARM Machine Directives

`.align` *expression* [, *expression*]

This is the generic *.align* directive. For the ARM however if the first argument is zero (ie no alignment is needed) the assembler will behave as if the argument had been 2 (ie pad to the next four byte boundary). This is for compatibility with ARM's own assembler.

`.arch` *name*

Select the target architecture. Valid values for *name* are the same as for the `-march` command-line option without the instruction set extension.

Specifying `.arch` clears any previously selected architecture extensions.

`.arch_extension` *name*

Add or remove an architecture extension to the target architecture. Valid values for *name* are the same as those accepted as architectural extensions by the `-mcpu` and `-march` command-line options.

`.arch_extension` may be used multiple times to add or remove extensions incrementally to the architecture being compiled for.

`.arm`

This performs the same action as *.code 32*.

`.bss`

This directive switches to the `.bss` section.

`.cantunwind`

Prevents unwinding through the current function. No personality routine or exception table data is required or permitted.

`.code [16|32]`

This directive selects the instruction set being generated. The value 16 selects Thumb, with the value 32 selecting ARM.

`.cpu` *name*

Select the target processor. Valid values for *name* are the same as for the `-mcpu` command-line option without the instruction set extension.

Specifying `.cpu` clears any previously selected architecture extensions.

name .dn *register name* [.*type*] [[*index*]]
name .qn *register name* [.*type*] [[*index*]]

The dn and qn directives are used to create typed and/or indexed register aliases for use in Advanced SIMD Extension (Neon) instructions. The former should be used to create aliases of double-precision registers, and the latter to create aliases of quad-precision registers.

If these directives are used to create typed aliases, those aliases can be used in Neon instructions instead of writing types after the mnemonic or after each operand. For example:

```
x .dn d2.f32
y .dn d3.f32
z .dn d4.f32[1]
vmul x,y,z
```

This is equivalent to writing the following:

```
vmul.f32 d2,d3,d4[1]
```

Aliases created using dn or qn can be destroyed using unreq.

.eabi_attribute *tag*, *value*

Set the EABI object attribute *tag* to *value*.

The *tag* is either an attribute number, or one of the following: Tag_CPU_raw_name, Tag_CPU_name, Tag_CPU_arch, Tag_CPU_arch_profile, Tag_ARM_ISA_use, Tag_THUMB_ISA_use, Tag_FP_arch, Tag_WMMX_arch, Tag_Advanced_SIMD_arch, Tag_MVE_arch, Tag_PCS_config, Tag_ABI_PCS_R9_use, Tag_ABI_PCS_RW_data, Tag_ABI_PCS_RO_data, Tag_ABI_PCS_GOT_use, Tag_ABI_PCS_wchar_t, Tag_ABI_FP_rounding, Tag_ABI_FP_denormal, Tag_ABI_FP_exceptions, Tag_ABI_FP_user_exceptions, Tag_ABI_FP_number_model, Tag_ABI_align_needed, Tag_ABI_align_preserved, Tag_ABI_enum_size, Tag_ABI_HardFP_use, Tag_ABI_VFP_args, Tag_ABI_WMMX_args, Tag_ABI_optimization_goals, Tag_ABI_FP_optimization_goals, Tag_compatibility, Tag_CPU_unaligned_access, Tag_FP_HP_extension, Tag_ABI_FP_16bit_format, Tag_MPextension_use, Tag_DIV_use, Tag_nodefaults, Tag_also_compatible_with, Tag_conformance, Tag_T2EE_use, Tag_Virtualization_use

The *value* is either a number, "string", or number, "string" depending on the tag.

Note - the following legacy values are also accepted by *tag*: Tag_VFP_arch, Tag_ABI_align8_needed, Tag_ABI_align8_preserved, Tag_VFP_HP_extension,

.even

This directive aligns to an even-numbered address.

.extend *expression* [, *expression*]*
.ldouble *expression* [, *expression*]*

These directives write 12byte long double floating-point values to the output section. These are not compatible with current ARM processors or ABIs.

.float16 *value [,...,value_n]*

Place the half precision floating point representation of one or more floating-point values into the current section. The exact format of the encoding is specified by `.float16_format`. If the format has not been explicitly set yet (either via the `.float16_format` directive or the command line option) then the IEEE 754-2008 format is used.

`.float16_format` *format*

Set the format to use when encoding float16 values emitted by the `.float16` directive. Once the format has been set it cannot be changed. `format` should be one of the following: `ieee` (encode in the IEEE 754-2008 half precision format) or `alternative` (encode in the Arm alternative half precision format).

`.fnend`

Marks the end of a function with an unwind table entry. The unwind index table entry is created when this directive is processed.

If no personality routine has been specified then standard personality routine 0 or 1 will be used, depending on the number of unwind opcodes required.

`.fnstart`

Marks the start of a function with an unwind table entry.

`.force_thumb`

This directive forces the selection of Thumb instructions, even if the target processor does not support those instructions

`.fpu` *name*

Select the floating-point unit to assemble for. Valid values for *name* are the same as for the `-mfpu` command-line option.

`.handlerdata`

Marks the end of the current function, and the start of the exception table entry for that function. Anything between this directive and the `.fnend` directive will be added to the exception table entry.

Must be preceded by a `.personality` or `.personalityindex` directive.

`.inst` *opcode* `[ , … ]`
`.inst.n` *opcode* `[ , … ]`
`.inst.w` *opcode* `[ , … ]`

Generates the instruction corresponding to the numerical value *opcode*. `.inst.n` and `.inst.w` allow the Thumb instruction size to be specified explicitly, overriding the normal encoding rules.

`.ldouble` *expression* `[,` *expression*`]*`

See `.extend`.

`.ltorg`

This directive causes the current contents of the literal pool to be dumped into the current section (which is assumed to be the .text section) at the current location (aligned to a word boundary). `GAS` maintains a separate literal pool for each section and each sub-section. The `.ltorg` directive will only affect the literal pool of the current section and sub-section. At the end of assembly all remaining, un-empty literal pools will automatically be dumped.

Note - older versions of `GAS` would dump the current literal pool any time a section change occurred. This is no longer done, since it prevents accurate control of the placement of literal pools.

`.movsp` *reg* `[, #`*offset*`]`

Tell the unwinder that *reg* contains an offset from the current stack pointer. If *offset* is not specified then it is assumed to be zero.

`.object_arch` *name*

Override the architecture recorded in the EABI object attribute section. Valid values for *name* are the same as for the `.arch` directive. Typically this is useful when code uses runtime detection of CPU features.

`.packed` *expression* `[,` *expression*`]*`

This directive writes 12-byte packed floating-point values to the output section. These are not compatible with current ARM processors or ABIs.

`.pad #`*count*

Generate unwinder annotations for a stack adjustment of *count* bytes. A positive value indicates the function prologue allocated stack space by decrementing the stack pointer.

`.personality` *name*

Sets the personality routine for the current function to *name*.

`.personalityindex` *index*

Sets the personality routine for the current function to the EABI standard routine number *index*

`.pool`

This is a synonym for .ltorg.

name `.req` *register name*

This creates an alias for *register name* called *name*. For example:

```
foo .req r0
```

`.save` *reglist*

Generate unwinder annotations to restore the registers in *reglist*. The format of *reglist* is the same as the corresponding store-multiple instruction.

core registers

```
  .save {r4, r5, r6, lr}
  stmfd sp!, {r4, r5, r6, lr}
```

FPA registers

```
  .save f4, 2
  sfmfd f4, 2, [sp]!
```

VFP registers

```
  .save {d8, d9, d10}
  fstmdx sp!, {d8, d9, d10}
```

iWMMXt registers

```
  .save {wr10, wr11}
  wstrd wr11, [sp, #-8]!
  wstrd wr10, [sp, #-8]!
or
  .save wr11
  wstrd wr11, [sp, #-8]!
  .save wr10
  wstrd wr10, [sp, #-8]!
```

`.setfp` *fpreg*, *spreg* [, *#offset*]

Make all unwinder annotations relative to a frame pointer. Without this the unwinder will use offsets from the stack pointer.

The syntax of this directive is the same as the `add` or `mov` instruction used to set the frame pointer. *spreg* must be either `sp` or mentioned in a previous `.movsp` directive.

```
.movsp ip
mov ip, sp
…
.setfp fp, ip, #4
add fp, ip, #4
```

`.secrel32` *expression* [, *expression*]*

This directive emits relocations that evaluate to the section-relative offset of each expression's symbol. This directive is only supported for PE targets.

`.syntax [unified | divided]`

This directive sets the Instruction Set Syntax as described in the ARM-Instruction-Set section.

`.thumb`

This performs the same action as *.code 16*.

`.thumb_func`

This directive specifies that the following symbol is the name of a Thumb

encoded function. This information is necessary in order to allow the assembler and linker to generate correct code for interworking between Arm and Thumb instructions and should be used even if interworking is not going to be performed. The presence of this directive also implies `.thumb`

This directive is not necessary when generating EABI objects. On these targets the encoding is implicit when generating Thumb code.

`.thumb_set`

This performs the equivalent of a `.set` directive in that it creates a symbol which is an alias for another symbol (possibly not yet defined). This directive also has the added property in that it marks the aliased symbol as being a thumb function entry point, in the same way that the `.thumb_func` directive does.

`.tlsdescseq` *tls-variable*

This directive is used to annotate parts of an inlined TLS descriptor trampoline. Normally the trampoline is provided by the linker, and this directive is not needed.

`.unreq` *alias-name*

This undefines a register alias which was previously defined using the `req`, `dn` or `qn` directives. For example:

```
foo .req r0
.unreq foo
```

An error occurs if the name is undefined. Note - this pseudo op can be used to delete builtin in register name aliases (eg 'r0'). This should only be done if it is really necessary.

`.unwind_raw` *offset*, *byte1*, …

Insert one of more arbitrary unwind opcode bytes, which are known to adjust the stack pointer by *offset* bytes.

For example `.unwind_raw 4, 0xb1, 0x01` is equivalent to `.save {r0}`

`.vsave` *vfp-reglist*

Generate unwinder annotations to restore the VFP registers in *vfp-reglist* using FLDMD. Also works for VFPv3 registers that are to be restored using VLDM. The format of *vfp-reglist* is the same as the corresponding store-multiple instruction.

VFP registers

```
.vsave {d8, d9, d10}
fstmdd sp!, {d8, d9, d10}
```

VFPv3 registers

```
.vsave {d15, d16, d17}
vstm sp!, {d15, d16, d17}
```

Since FLDMX and FSTMX are now deprecated, this directive should be used in favour of `.save` for saving VFP registers for ARMv6 and above.

Next: ARM Opcodes, Previous: ARM Floating Point, Up: ARM-Dependent
[Contents][Index]