Directives

R. Ferrero

Politecnico di Torino

Dipartimento di Automatica e Informatica (DAUIN)

Torino - Italy

Instruction format

- A general source line is:

`{label} {operation} {;comment}`

- `operation` may be:
  - an instruction
  - a pseudo-instruction
  - a directive.
- A line may have up to 4095 characters.
- A line can be split into several lines by placing a backslash character (\) at the end.

Common directives

- AREA
- RN
- EQU
- DCB, DCW, DCWU, DCD, DCDU
- ALIGN
- SPACE
- LTORG
- PROC/ENDP o FUNCTION/ENDFUNC
- END

Sections of data and code

- `AREA` *`sectionName`* `{,`*`attr`*`} {,`*`attr`*`}…`
- If *`sectionName`* starts with a number, it must be enclosed in bars
  e.g. `|1_DataArea|`
- `|.text|` is used by the C compiler
- At least one `AREA` directive is mandatory
- Example: `AREA Example,CODE,READONLY`

Section attributes

- `CODE`: the section contains machine code
- `DATA`: the section contains data
- `READONLY`: the section can be placed in read-only memory
- `READWRITE`: the section can be placed in read-write memory
- `ALIGN = expr`: the section is aligned on a 2^{expr} byte boundary

Definition of new register names

- `RN` defines the alias of a register:

  `name` `RN` *`registerIndex`*

- Example: `coeff1 RN 8`
- *`registerIndex`* ranges between 0 and 15.
- Using `RN` is not mandatory, but it increases code readability.

Declaring constants

- The `EQU` directive gives a symbolic name to a numeric constant:

  `name` `EQU` *`expression`*

- Example: `COLUMNS EQU 8`
- Advantages:
  - readibility
  - easiness in updating the value through the code

Numbers

- You can express numbers in any base:
  - decimal: e.g. 123
  - hexadecimal: e.g. 0x3F
  - other bases in the format n_xxx where
    - n is the base between 2 and 9
    - xxx is the number in that base

Characters and strings

- A character is written between single quotes.
  - example: 'a'
- Escape characters are written like in C.
  - example: new line is '\n'

- A string is written between double quotes.
  - example: "Hello world!"
  - Strings are not null-terminated.

Memory allocation

{*label*} DC*xx* *expr*{,*expr*}…

- The available directives are:
  - DCB: define constant byte
  - DCW: define constant half-word
  - DCWU: define constant half-word unaligned
  - DCD: define constant word
  - DCDU: define constant word unaligned
- *expr* is:
  - a numeric expression in the proper range
  - a string (with DCB only)

DCB

```
    AREA example, DATA, READONLY
myData  DCB 65, 0x73, 8_163
        DCB "embly"
```

Address	Value	Octal	Hex	ASCII
0x000000D2	65	101	41	A
0x000000D3	115	163	73	s
0x000000D4	115	163	73	s
0x000000D5	101	145	65	e
0x000000D6	109	155	6D	m
0x000000D7	98	142	62	b
0x000000D8	108	154	6C	l
0x000000D9	121	171	79	y

DCW

```
    AREA example, DATA, READONLY
myData  DCB 65, 0x73, 8_163
        DCW 0x626D, 0x796C
```

Address	Value	Octal	Hex	ASCII
0x000000D2	65	101	41	A
0x000000D3	115	163	73	s
0x000000D4	115	163	73	s
0x000000D5	0	0	0	NUL
0x000000D6	109	155	6D	m
0x000000D7	98	142	62	b
0x000000D8	108	154	6C	l
0x000000D9	121	171	79	y

DCWU

```
    AREA example, DATA, READONLY
myData  DCB 65, 0x73, 8_163
        DCWU 0x626D, 0x796C
```

Address	Value	Octal	Hex	ASCII
0x000000D2	65	101	41	A
0x000000D3	115	163	73	s
0x000000D4	115	163	73	s
0x000000D5	109	155	6D	m
0x000000D6	98	142	62	b
0x000000D7	108	154	6C	l
0x000000D8	121	171	79	y

DCD

```
    AREA example, DATA, READONLY
myData  DCB 65, 0x73, 8_163
        DCD 0x796C626D
```

Address	Value	Octal	Hex	ASCII
0x000000D2	65	101	41	A
0x000000D3	115	163	73	s
0x000000D4	115	163	73	s
0x000000D5	0	0	0	NUL
0x000000D6	0	0	0	NUL
0x000000D7	0	0	0	NUL
0x000000D8	109	155	6D	m
0x000000D9	98	142	62	b
0x000000DA	108	154	6C	l
0x000000DB	121	171	79	y

DCDU

```
    AREA example, DATA, READONLY
myData  DCB 65, 0x73, 8_163
        DCDU 0x796C626D
```

Address	Value	Octal	Hex	ASCII
0x000000D2	65	101	41	A
0x000000D3	115	163	73	s
0x000000D4	115	163	73	s
0x000000D5	109	155	6D	m
0x000000D6	98	142	62	b
0x000000D7	108	154	6C	l
0x000000D8	121	171	79	y

Aligning data or code

- The `ALIGN` directive aligns the current location to a specified boundary by padding with zeros:

  `ALIGN {`*`expr`*`{, `*`offset`*`}}`

- The current location is aligned to the next address of the form

  n * *`expr`* + *`offset`*

- If *`expr`* is not specified, `ALIGN` sets the current location to the next word boundary.

ALIGN expr

```
        AREA example, DATA, READONLY
myData  DCB 65
        ALIGN 2
        DCB 115
```

Address	Value	Octal	Hex	ASCII
0x000000D4	65	101	41	A
0x000000D5	0	0	0	NUL
0x000000D6	115	163	73	s

ALIGN expr

```
    AREA example, DATA, READONLY
myData  DCB 65
        ALIGN 4
        DCB 115
```

Address	Value	Octal	Hex	ASCII
0x000000D4	65	101	41	A
0x000000D5	0	0	0	NUL
0x000000D6	0	0	0	NUL
0x000000D7	0	0	0	NUL
0x000000D8	115	163	73	s

ALIGN expr, offset

```
        AREA example, DATA, READONLY
myData  DCB 65
        ALIGN 4, 3
        DCB 115
```

Address	Value	Octal	Hex	ASCII
0x000000D4	65	101	41	A
0x000000D5	0	0	0	NUL
0x000000D6	0	0	0	NUL
0x000000D7	115	163	73	s

Reserving a block of memory

- The `SPACE` directive reserves a zeroed block of memory:

`{`*`label`*`} SPACE` *`expr`*

- *`expr`* is the number of bytes to reserve
- Example:

```
    AREA example, DATA, READWRITE
word_var SPACE 4
halfword_var SPACE 2
```

Assigning literal pool origins

- Literal pools are constants that can not be assigned to a register with a `MOV` instruction.
- By default, literal pools are put at the end of code sections.
- The `LTORG` directive forces the assembler to put a literal pool within the code section.

Start and end of a function

- Only a label is required to call a function:

```
my_func ADDS r0,r0,r1 ;add 2 params
        BX LR        ;return
```

- To improve clarity, directives can be added to indicate start and end of the function.

```
my_func PROC         ;or FUNCTION
        ADDS r0,r0,r1 ;add 2 params
        BX LR        ;return
        ENDP         ;or ENDFUNC
```

Ending the source file

- The `END` directive tells the assembler that the current location is the end of the source file:

```
END
```