CPUASM 2.00 (TM)

gferrante@opencores.org

Overview

Usage: **cpuasm** -a ADDRESS_CODE -b SIZE_CODE -c ADDRESS_DATA -d SIZE_DATA -p CPU_ID -s IST_SET_FILE -i INPUT_ASM_FILE -o OUTPUT_DIRECTORY

(ex.) cpuasm -a 10 -b 14 -c 7 -d 8 -p 0 -s cpu.ist -i soc.asm -o sw_target_directory

Input files: assembler file, instruction set file, target software directory

Output files:

- rom#.mif: rom file used by Altera rom
- ram#.mif: ram file used by Altera ram
- rom#.coe: rom file used by Xilinx rom
- ram#.coe: ram file used by Xilinx ram
- rom#.bin: rom file used for binary upload
- ram#.bin: ram file used for binary upload
- rom#.vin: rom file used by VHDL simulator (TestBench)
- ram#.vin: ram file used by VHDL simulator (TestBench)

Language

COMMENT: --

ZONE:

```
__equ;      -- Define zone
__rom;      -- Code zone
__ram;      -- Ram and I/O registers zone
__reg;      -- I/O registers zone
```

VALUE:

```
#[value]          -- Integer
#[value]h         -- Hex
#[value]b         -- Binary
#[value]c         -- Character
#[string]s        -- String              (only in __ram zone)
#[ivector]v       -- Vector of integer   (only in __ram zone)
```

Note: "[" and "]" haven't to be printed.

STRING:

char_0...char_n<integer>char_n+1...

ex. Hello World => #Hello<20>World<13>s

Note: "<" and ">" have to be printed.

IVECTOR:

<integer_0>...<integer_n>

ex. 1, -2, 3 => #<1><-2><3>v

Note: "<" and ">" have to be printed.

REPEAT:

```
*[value]          -- Integer  (Valid only in __ram and __reg zone)
*[value]h         -- Hex      (Valid only in __ram and __reg zone)
*[value]b         -- Binary   (Valid only in __ram and __reg zone)
*[value]c         -- Character(Valid only in __ram and __reg zone)
```

Note: "[" and "]" haven't to be printed.

ADDR_VARIABLE:

```
&VARIABLE         -- Address of variable  (Valid only in __rom zone)
```

FLAGS: ZERO (Z), CARRY (C); -- Zero => Z = 1; CARRY => C = 1

PC: PROGRAM COUNTER

STACK: ADDRESS STACK -- Internal or external

ADDR: label defined in __rom zone

DEFINE: label defined in __equ zone

```
label_define CONSTANT;  -- NOTE: no ':' after label define;
```

CONSTANT:

```
- VALUE
- label_define
```

SET CURRENT ADDRESS:

```
__org CONSTANT; -- Zone dependent
```

VARIABLE: label defined in __ram or __reg zone

```
label:      ;           -- define a ram/reg variable initialized to 0
label:  CONSTANT;       -- define a ram/reg variable initialized to
                        -- CONSTANT
label:  *REPEAT;        -- define a ram/reg vector initialized to 0 of
                        -- size REPEAT
label:  CONSTANT *REPEAT;     -- define a ram/reg vector initialized to
                              -- CONSTANT of size REPEAT
```

NOTE: variables are case sensitive, instructions are not case sensitive

Instruction set file:

Instruction set file permits to customize cpuasm behaviour, the cpu vhdl encoding must be changed by directly editing the files generated by cpucore (UTILS, CU, DU).

Cpuasm encoding:

MSB LSB
SIZE_CODE-1 INIT_ARG_1 | 0
ARGUMENT1 | ISTRUCTION_CODE

-- Comment
--
-- Assembler instructions format
--
ISTRUCTION_NAME NUMBER_OF_ARGUMENTS [NOA]
ISTRUCTION_CODE INIT_ARG_1 … INIT_ARG_NOA

Ex:

```
ldai  1
A     4
nop   0
00
```

NOTE:

- supported only NUMBER_OF_ARGUMENTS = (0,1)
- numbers in hexadecimal radix

Default instruction set: (4 bit encoding)

1) **addi** **COSTANT;** -- add accumulator with constant

- accumulator <= accumulator + CONSTANT
- modify (Z,C)
- encoding |COSTANT|1100|
- class (TC_DAT, TD_ADD)

2) **add** **VARIABLE;** -- add accumulator with variable content

- accumulator <= accumulator + (VARIABLE)
- modify (Z,C)
- encoding |VARIABLE|0100|
- class (TC_LDA, TD_ADD)

3) **andi** **COSTANT;** -- and accumulator with constant

- accumulator <= accumulator and CONSTANT
- modify (Z, C = 0)
- encoding |COSTANT|1101|
- class (TC_DAT, TD_AND)

4) **and** **VARIABLE;** -- and accumulator with variable content

- accumulator <= accumulator and (VARIABLE)
- modify (Z, C = 0)
- encoding |VARIABLE|0101|
- class (TC_LDA, TD_AND)

5) **subi** **COSTANT;** -- subtract accumulator with constant

- accumulator <= accumulator - CONSTANT
- modify (Z,C)
- encoding |COSTANT|1110|
- class (TC_DAT, TD_SUB)

6) **sub** **VARIABLE;** -- subtract accumulator with variable content

- accumulator <= accumulator - (VARIABLE)
- modify (Z,C)
- encoding |VARIABLE|0110|
- class (TC_LDA, TD_SUB)

7) **ori** **CONSTANT;** -- or accumulator with constant

- accumulator <= accumulator or CONSTANT
- modify (Z, C = 0)
- encoding |COSTANT|1111|
- class (TC_DAT, TD_OR)

8) **or** **VARIABLE;** -- or accumulator with variable content

- accumulator <= accumulator or (VARIABLE)
- modify (Z, C = 0)
- encoding |VARIABLE|0111|
- class (TC_LDA, TD_OR)

9) **xori** **CONSTANT;** -- xor accumulator with constant

- accumulator <= accumulator xor CONSTANT
- modify (Z, C = 1)
- encoding |COSTANT|1011|
- class (TC_DAT, TD_XOR)

10) **xor** **VARIABLE;** -- xor accumulator with variable content

- accumulator <= accumulator xor (VARIABLE)
- modify (Z, C = 1)
- encoding |VARIABLE|0011|
- class (TC_LDA, TD_XOR)

11) **ldai** **CONSTANT;** -- load accumulator with constant

- accumulator <= CONSTANT
- modify (Z, C = 0)
- encoding |COSTANT|1010|
- class (TC_DAT, TD_LDA)

12) **lda** **VARIABLE;** -- load accumulator with variable content

- accumulator <= (VARIABLE)
- modify (Z, C = 0)
- encoding |VARIABLE|0010|
- class (TC_LDA, TD_LDA)

13) **sta** **VARIABLE;** -- store accumulator

- (VARIABLE) <= accumulator
- modify ()
- encoding |VARIABLE|0001|
- class (TC_STA, TD_NOP)

14) **jmp** **ADDR;** -- jump

- PC <= ADDR
- modify ()
- encoding |ADDR|1000|
- class (TC_JMP, TD_NOP)

15) **jms** **ADDR;** -- jump to subroutine

- STACK <= PC; PC <= ADDR
- modify ()
- encoding |ADDR|1001|
- class (TC_JMS, TD_NOP)

16) **rts;** -- return from subroutine

- PC <= STACK
- modify ()
- encoding |XXXX1100|0000|
- class (TC_RTS, TD_NOP)

17) **rti;** -- return from interrupt

- PC <= STACK
- modify ()
- encoding |XXXX1101|0000|
- class (TC_RTI, TD_NOP)

18) **nop;** -- no operation

- modify ()
- encoding |XXXX0000|0000|
- class (TC_OTH, TD_NOP)

19) **rol;** -- rotate left accumulator with carry

- accumulator <= rotate_left(CARRY + accumulator)
- modify (Z,C)

- encoding |XXXX0001|0000|

- class (TC_OTH, TD_ROL)

20) **ror;** -- rotate right accumulator with carry

- accumulator <= rotate_right(CARRY + accumulator)

- modify (Z,C)

- encoding |XXXX0010|0000|

- class (TC_OTH, TD_ROR)

21) **shl;** -- shift left accumulator with carry

- accumulator <= shift_left(CARRY + accumulator) + '0'

- modify (Z,C)

- encoding |XXXX0011|0000|

- class (TC_OTH, TD_SHL)

22) **shr;** -- shift right accumulator with carry

- accumulator <= '0' + shift_right(CARRY + accumulator)

- modify (Z,C = 0)

- encoding |XXXX0100|0000|

- class (TC_OTH, TD_SHR)

23) **not;** -- not accumulator

- accumulator <= not (accumulator)

- modify (Z,C = 0)

- encoding |XXXX0101|0000|

- class (TC_OTH, TD_NOT)

24) **cla;** -- clear accumulator

- accumulator <= 0

- modify (Z = 1,C = 0)

- encoding |XXXX0110|0000|

- class (TC_OTH, TD_CLA)

25) **skz;** -- skip one istruction if zero

- PC <= (PC + 2) if (Z = 1) else (PC + 1)

- modify (C = 0)

- encoding |XXXX1000|0000|

- class (TC_OTH, TD_SKZ)

26) **sknz;** -- skip one istruction if not zero

- PC <= (PC + 2) if (Z = 0) else (PC + 1)
- modify (C = 0)
- encoding |XXXX1010|0000|
- class (TC_OTH, TD_SKNZ)

27) **skc;** -- skip one istruction if carry

- PC <= (PC + 2) if (C = 1) else (PC + 1)
- modify (C = 0)
- encoding |XXXX1001|0000|
- class (TC_OTH, TD_SKC)

28) **sknc;** -- skip one istruction if not carry

- PC <= (PC + 2) if (C = 0) else (PC + 1)
- modify (C = 0)
- encoding |XXXX1011|0000|
- class (TC_OTH, TD_SKNC)

NOTE:

- Maximum SIZE_CODE = 32
- Maximum SIZE_DATA = 32
- The instructions "cla" and "lda #0" can operate differently if data expansion register size > 0 (see cpu hardware architecture)
- # = CPU_ID (0 = no value)