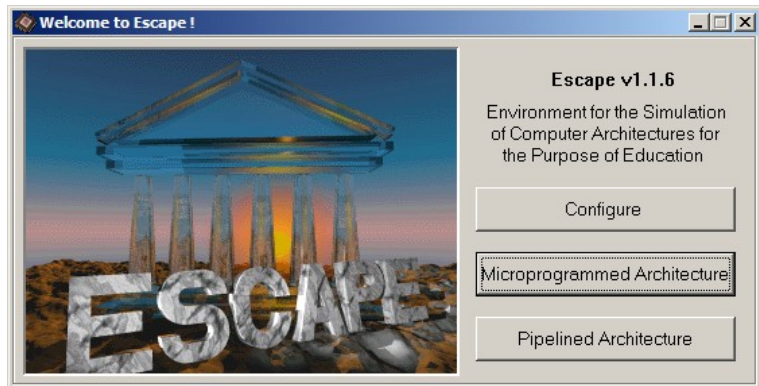




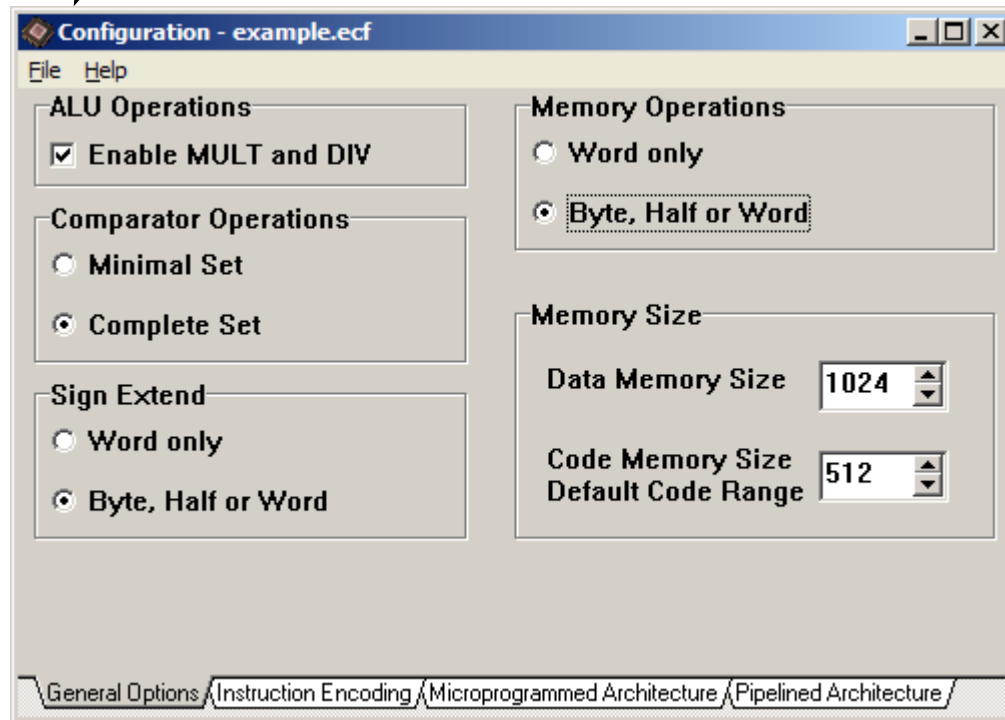
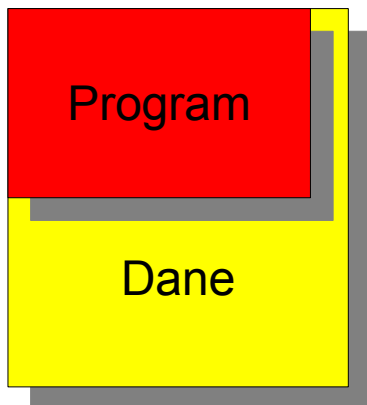
Symulator Escape

Konfiguracja ogólna

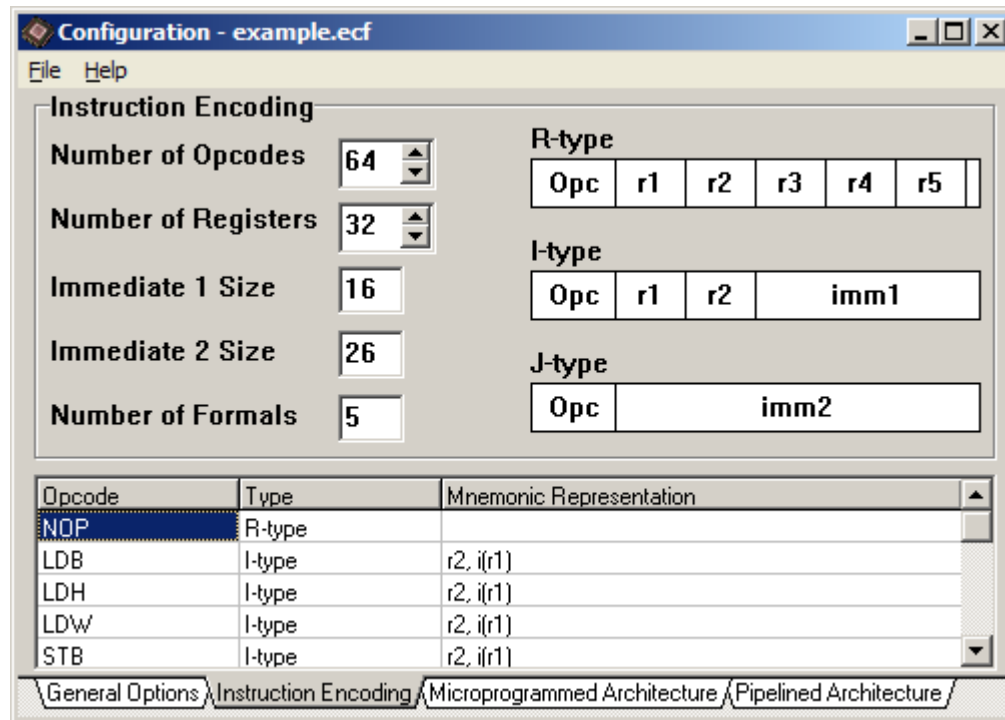


Załaduj konfigurację symulatora
(File -> OpenFile)
z pliku **example.ecf**

Enable MUL and DIV
Complete Set of Comp. Oper
Sign Extension of B/H/W
Memory Oper on B/H/W



Lista i format instrukcji



64 typy instr. → 6 bitowe pole typu instrukcji (*opcode*)

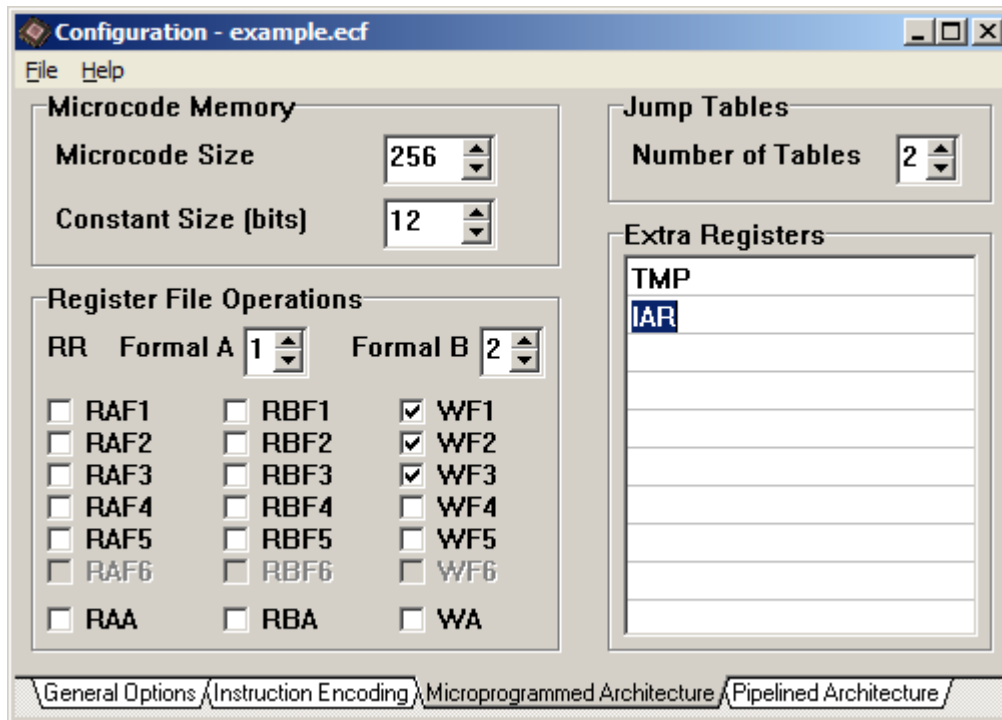
32 rejestry wewn. → 5 bitowe numery rejestrów (*formal fields: r1,r2,...*), R0 ... R31

Stałe w kodzie instrukcji (*immediate*):

imm1 – 16 bitów dla instrukcji Load/Store oraz skoków warunkowych (Bxx)

imm2 – 26 bitów dla instrukcji skoku bezwarunkowego (JMP)

Obsługa pliku rejestrów i mikrokod



Pamięć mikro kodu → 256 12-bitowych słów

Operacje na pliku rejestrów:

odczyt: *RR* (*Read Registers*) → odczyt do A i B zaw. rej. o numerze z pól *r1* i *r2*

zapis: *WF1*, *WF2*, *WF3* (*Write Formal n*) → zapis do rejestru o numerze z pola *r1*, *r2*, *r3*

Tablice skoków dla sterowania mikro kodem → 2

Rejestry dodatkowe (niezależnie od rejestrów R0..R31)

Projekt

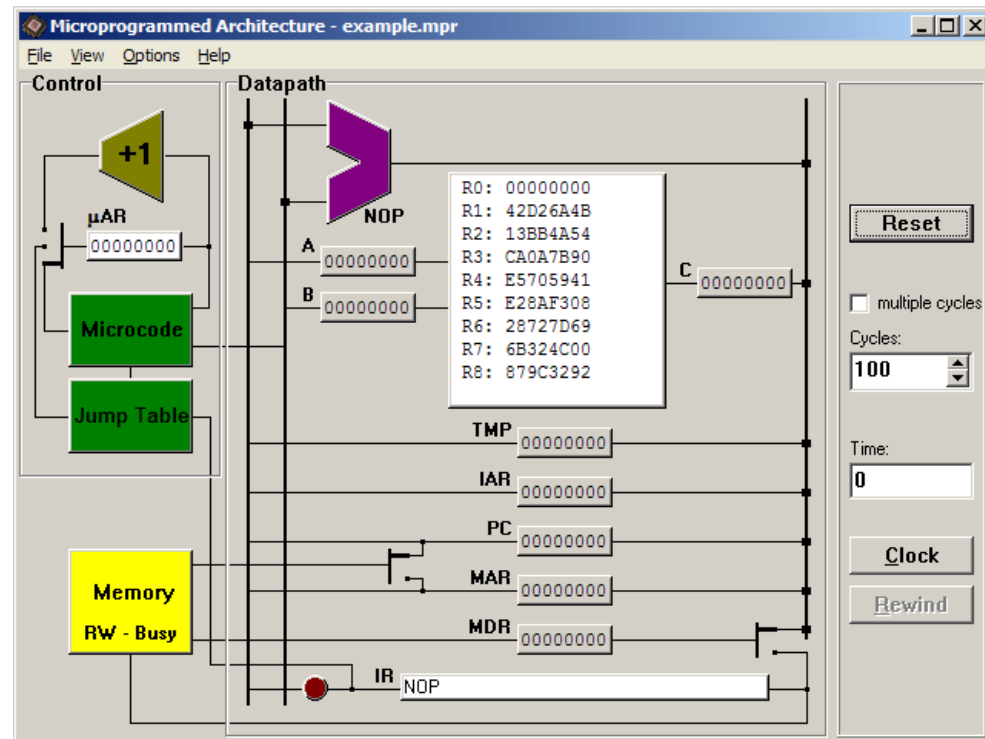
Załaduj projekt
(File -> OpenFile)
z pliku **example.mpr**

Projekt
(.mpr – microcode project)

Program w asemblerze
(.cod - code)

Mikrokod
(.mco - microcode)

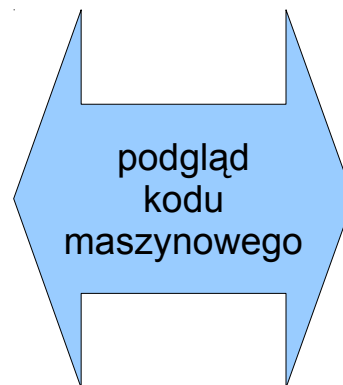
Zawartość pamięci
(.dat - data)



Podgląd pamięci

```

Memory (Instruction View) - EXAMPLE.COD
File Edit View Help
0000: 4401000A   ADDI R0, 0x000A, R1
0004: 10210200   loop1  STB  R1, 0x0200 (R1)
0008: 48210001   SUBI R1, 0x0001, R1
000C: 7801FFFF   BRGT R1, loop1
0010: 00000000   NOP
0014: 00000000   NOP
0018: 00000000   NOP
001C: 00000000   NOP
0020: 34421000   XOR  R2, R2, R2
0024: 4401000A   ADDI R0, 0x000A, R1
0028: 04230200   loop2  LDB  R3, 0x0200 (R1)
002C: 1C431000   ADD  R2, R3, R2
0030: 48210001   SUBI R1, 0x0001, R1
0034: 7801FFFF   BRGT R1, loop2
0038: 00000000   NOP
003C: 00000000   NOP
0040: 00000000   NOP
0044: 7000FFFC   halt  BRZ  R0, halt
0048: 00000000   NOP
    
```



```

Memory (Data View) - EXAMPLE.DAT
File Edit View Help
000: 4401000A 10210200 48210001 7801FFF4
010: 00000000 00000000 00000000 00000000
020: 34421000 4401000A 04230200 1C431000
030: 48210001 7801FFFF 00000000 00000000
040: 00000000 7000FFFC 00000000 00000000
050: 00000000 00000000 00000000 00000000
060: 00000000 00000000 00000000 00000000
070: 00000000 00000000 00000000 00000000
080: 00000000 00000000 00000000 00000000
090: 00000000 00000000 00000000 00000000
0A0: 00000000 00000000 00000000 00000000
0B0: 00000000 00000000 00000000 00000000
0C0: 00000000 00000000 00000000 00000000
0D0: 00000000 00000000 00000000 00000000
0E0: 00000000 00000000 00000000 00000000
0F0: 00000000 00000000 00000000 00000000
100: 00000000 00000000 00000000 00000000
110: 00000000 00000000 00000000 00000000
120: 00000000 00000000 00000000 00000000
130: 00000000 00000000 00000000 00000000
140: 00000000 00000000 00000000 00000000
150: 00000000 00000000 00000000 00000000
    
```

```

Memory (Data View) - EXAMPLE.DAT
File Edit View Help
000: 44 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
010: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
020: 34 42 10 00 00 00 00 00 00 00 00 00 00 00 00 00
030: 48 21 00 01 78 01 FF F0 00 00 00 00 00 00 00 00
040: 00 00 00 00 70 00 FF FC 00 00 00 00 00 00 00 00
050: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
060: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
070: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
080: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
    
```

Możliwość wybór rozmiaru (B/H/W) oraz kodowania (hex, un-, signed)

```

Memory (Data View) - EXAMPLE.DAT
File Edit View Help
0: 44 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
10: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
20: 34 42 10 00 00 00 00 00 00 00 00 00 00 00 00 00
30: 48 21 00 01 78 01 FF F0 00 00 00 00 00 00 00 00
40: 00 00 00 00 70 00 FF FC 00 00 00 00 00 00 00 00
50: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
60: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
70: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
80: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
    
```

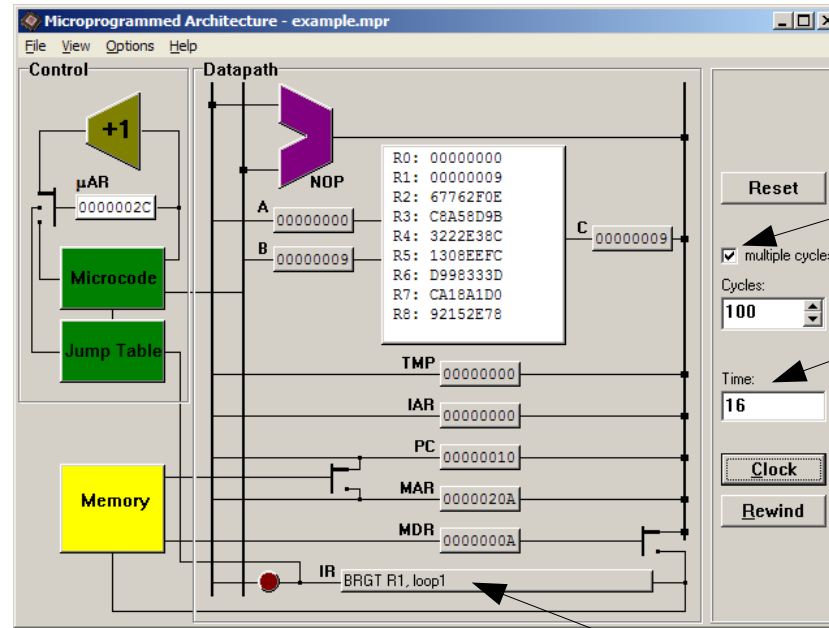
Praca krokowa i pułapki

```

Memory (Instruction View) - EXAMPLE.COD
File Edit View Help
0000: 4401000A ADDI R0, 0x000A, R1
0004: 10210200 loop1 STB R1, 0x0200 (R1)
0008: 48210001 SUBI R1, 0x0001, R1
000C: 7801FFF4 BRGT R1, loop1
0010: 00000000 NOP
001: 00000000 NOP
0018: 00000000 NOP
Range: 0000-01FC Overwrite
    
```

ustawienie pułapki na wartość 10 dla PC

! Uwaga: pułapka na np. PC=10 powoduje zatrzymanie programu gdy w PC pojawi się wartość 10, ale nie oznacza to, że instrukcja spod adresu 10 została wykonana



Zezwolenie na wykonanie wielu cykli zegara na raz

Czas (liczba cykli) działania programu

Wymuszenie 1 lub wielu cykli zegara

Aktualnie wykonywana (pobrana) instrukcja

```

Breakpoints
View Help
Organisational Registers
[ ] uAR = 0x00000000
[ ] A = 0x00000000
[ ] B = 0x00000000
[ ] C = 0x00000000
[ ] PC = 0x00000010
Register File Registers
[ ] R 1 = 0x00000000
[ ] R 2 = 0x00000000
[ ] R 3 = 0x00000000
[ ] R 4 = 0x00000000
[ ] R 5 = 0x00000000
    
```

```

Note
i Breakpoint condition met
OK
    
```

Instrukcje transferu

LDB	Ry, i (Rx)	Load Byte:	Ry.B ← Mem[Rx+i]
LDH	Ry, i (Rx)	Load Half:	Ry.H ← Mem[Rx+i]
LDW	Ry, i (Rx)	Load Word:	Ry.W ← Mem[Rx+i]
STB	Ry, i (Rx)	Store Byte:	Mem[Rx+i] ← Ry.B
STH	Ry, i (Rx)	Store Half:	Mem[Rx+i] ← Ry.H
STW	Ry, i (Rx)	Store Word:	Mem[Rx+i] ← Ry.W

Format instrukcji LD i ST



- Rx,Ry,Rz** – rejestry wewnętrzne procesora: R0...R31
- i** – stała (liczba) 16-bitowa (2B), podczas operacji rozszerzana znakowo do 32-bitów
- label** – etykieta linii programu w assemblerze (adres skoku)
- .B** – bajt (8-bitów), Rx.B - najmniej znaczący bajt rejestru
- .H** – półsłowo (16-bitów), Rx.H - mniej znaczące półsłowo rejestru
- .W** – słowo (32-bity), Rx.W – cała zawartość 32-bitowego rejestru

Instrukcje arytmetyczne i logiczne

ADD	Rx, Ry, Rz	Add:	Rz ← Rx+Ry
SUB	Rx, Ry, Rz	Sub:	Rz ← Rx-Ry
MUL	Rx, Ry, Rz	Mul:	Rz ← Rx*Ry
DIV	Rx, Ry, Rz	Div:	Rz ← Rx/Ry
ADDI	Rx, i, Ry	Add:	Ry ← Rx+i
SUBI	Rx, i, Ry	Add:	Ry ← Rx-i
MULI	Rx, i, Ry	Add:	Ry ← Rx*i
DIVI	Rx, i, Ry	Add:	Ry ← Rx/i
AND	Rx, Ry, Rz	And:	Rz ← Rx&Ry
OR	Rx, Ry, Rz	Or :	Rz ← Rx Ry
XOR	Rx, Ry, Rz	Xor:	Rz ← Rx^Ry
ANDI	Rx, i, Ry	And:	Ry ← Rx&i
ORI	Rx, i, Ry	Or :	Ry ← Rx i
XORI	Rx, i, Ry	Xor:	Ry ← Rx^i
SLL	Rx, Ry, Rz	Shift Left Logical:	Rz ← Rx<<Ry
SRL	Rx, Ry, Rz	Shift Right Logical:	Rz ← Rx>>Ry
SRA	Rx, Ry, Rz	Shift Right Arithm.:	Rz ← Rx>>Ry with sign
SLLI	Rx, i, Ry	Shift Left Logical:	Ry ← Rx<<i
SRLI	Rx, i, Ry	Shift Right Logical:	Ry ← Rx>>i
SRAI	Rx, i, Ry	Shift Right Arithm.:	Ry ← Rx>>i with sign

Instrukcje skoków i pozostałe

Dla wszystkich skoków warunkowych:

BRcc: $PC \leftarrow \text{label}(PC + \text{offset})$ if condition(cc) == True

BRZ	Rx, label	Branch if Zero	: condition Rx = 0
BRNZ	Rx, label	Branch if Not Zero	: condition Rx \neq 0
BRGT	Rx, label	Branch if Greater Than	: condition Rx > 0
BRGE	Rx, label	Branch if Greater or Equal	: condition Rx \geq 0
BRLT	Rx, label	Branch if Less Than	: condition Rx < 0
BRLE	Rx, label	Branch if Less or Equal	: condition Rx \leq 0

Inne:

NOP No Operation

LIH Rx, i Load Immediate High: $Rx[31..16] \leftarrow i.H$

Tryby adresowania

Tryby adresowania są sposobem na zapisywanie lokalizacji operandów, biorących udział w operacji

Adresowanie natychmiastowe (*Immediate*) – operand w kodzie instrukcji

ADDI R5, 0x55AA, R3

Adresowanie rejestrowe bezpośrednio (*Register Direct*) – operand w rejestrze

LDW R7, 0x20 (R3)

Adresowanie rejestrowe pośrednie z przesunięciem (*Register Indirect with Displacement*) - operand w komórce pamięci, której adres jest w rejestrze (+ przesunięcie)

Assembler – przykład: n!

R1 – wejściowa wartość n
R2 – wynik

! Transfery pomiędzy rejestrami realizowane są za pomocą dodawania R0 (który ma zawsze wartość zero)

Metoda iteracyjna: $n! = n*(n-1)*(n-2)*...*1$

```
ADDI R0,0x000A,R1
ADD R1,R0,R2
next SUBI R1,0x0001,R1
BRZ R1, halt
MUL R2,R1,R2
BRZ R0,next
halt BRZ R0, halt
```

ładowanie n=10 do R1
ładowanie do R2 wartość początkowej (n)
dekrementacja n ($R1 \leftarrow R1-1$)
skok gdy n=0 (koniec obliczeń)
mnożenie wyniku przez n-1 ($R2 \leftarrow R2*R1$)
skok bezwarunkowy (następny cykl obliczeń)
stop

etykieta mnemonik operandy (zapisane za pomocą różnych trybów adresowania)

```
Memory (Instruction View) - EXAMPLE.COD
File Edit View Help
0000: 4401000A ADDI R0, 0x000A, R1
0004: 1C201000 ADD R1, R0, R2
0008: 48210001 next SUBI R1, 0x0001, R1
000C: 70010008 BRZ R1, halt
0010: 24411000 MUL R2, R1, R2
0014: 7000FFFO BRZ R0, next
0018: 7000FFFC halt BRZ R0, halt
001C: 00000000 NOP
Range: 0000-01FC Overwrite
```