# C: Array and String

| op | name | cycles | adrmode | flags |
|---|---|---|---|---|
| 0x01 | BRK | 1 | imp | NVBDIZC -1-1- |

description

interrupt

function

(S)-=:PC,P PC:=($FFFE)

logic

```
PC = PC + 1
bPoke(SP,PC.h)
SP = SP − 1
bPoke(SP,PC.l)
SP = SP − 1
bPoke(SP, (P|\$10) )
SP = SP − 1
l = bPeek(\$FFFE)
h = bPeek(\$FFFF)<<8
PC = h|l
```

Example

BRK

BRK

| op | name | cycles | adr | flags |
|---|---|---|---|---|
| 0x00 | BRK | 1 | imp | NVBDIZC -1-1- |

description

interrupt

function

(S)-=:PC,P PC:=($FFFE)

logic

```
PC = PC + 1
bPoke(SP,PC.h)
SP = SP − 1
bPoke(SP,PC.l)
SP = SP − 1
bPoke(SP, (P|\$10) )
SP = SP − 1
l = bPeek(\$FFFE)
h = bPeek(\$FFFF)<<8
PC = h|l
```

example

BRK

| op | name | adr | bytes | cycles | function |
|---|---|---|---|---|---|
| 0x01 | ORA | izx | 2 | 2 | A:=A or adr |

flags

NVBDIZC
*-----*-

description

or with accumulator

logic

```
A = A | M
P.N = A.7
P.Z = (A==0) ? 1:0
```

| example | c–fct |
|---|---|
| ORA ($A5,X) | cpu_6502_ora_izy |

<table>
<tr><th>opcode</th><th>mnmonic</th><th>bytes</th><th>cycles</th><th>adr"-mode</th><th>function</th><th>flags</th><th>logic</th><th>example</th></tr>
<tr><td>0x00</td><td>BRK</td><td>1</td><td>0</td><td>imp</td><td>interrupt<br>(S)-=:PC,P<br>PC:=($FFFE)</td><td>NVBDIZC<br>-1-1-</td><td>PC = PC + 1<br>bPoke(SP,PC.h)<br>SP = SP − 1<br>bPoke(SP,PC.l)<br>SP = SP − 1<br>bPoke(SP, (P|\$10) )<br>SP = SP − 1<br>l = bPeek(\$FFFE)<br>h = bPeek(\$FFFF)<<8<br>PC = h|l</td><td>BRK</td></tr>
<tr><td>| **Text1** |</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>0x01</td><td>ORA</td><td>2</td><td>6</td><td>izx</td><td>or with accumulator<br>A:=A or adr</td><td>NVBDIZC<br>*--*-</td><td>A = A | M<br>P.N = A.7<br>P.Z = (A==0) ? 1:0</td><td>ORA ($A5,X)</td></tr>
<tr><td>0x05</td><td>ORA</td><td>2</td><td>3</td><td>zp</td><td>or with accumulator<br>A:=A or adr</td><td>NVBDIZC<br>*--*-</td><td>A = A | M<br>P.N = A.7<br>P.Z = (A==0) ? 1:0</td><td>ORA $AB</td></tr>
</table>

Table 1: opcode table

<table>
<tr><td>op<br>0x00</td><td>name<br>BRK</td><td>adr<br>imp</td><td>bytes<br>1</td><td>cycles<br>0</td><td>function<br>(S)-=:PC,P PC:=($FFFE)</td></tr>
<tr><td colspan="4">flags<br>NVBDIZC<br>--1-1--</td><td colspan="2" rowspan="2">logic<br>PC = PC + 1<br>bPoke(SP,PC.h)<br>SP = SP − 1<br>bPoke(SP,PC.l)<br>SP = SP − 1<br>bPoke(SP, (P|$10) )<br>SP = SP − 1<br>l = bPeek($FFFE)<br>h = bPeek($FFFF)<<8<br>PC = h|l</td></tr>
<tr><td colspan="4">description<br>interrupt</td></tr>
<tr><td colspan="4">example<br>BRK</td><td colspan="2">c–fct<br>void cpu_6502_BRK_imp();</td></tr>
</table>

<table>
<tr><td>op<br>0x00</td><td>name<br>BRK</td><td>adr<br>imp</td><td colspan="2">NVBDIZC<br>--1-1--</td><td rowspan="5">logic<br>PC = PC + 1<br>bPoke(SP,PC.h)<br>SP = SP − 1<br>bPoke(SP,PC.l)<br>SP = SP − 1<br>bPoke(SP, (P|$10) )<br>SP = SP − 1<br>l = bPeek($FFFE)<br>h = bPeek($FFFF)<<8<br>PC = h|l</td></tr>
<tr><td colspan="3">example<br>BRK</td><td>bytes<br>1b</td><td>cycles<br>1c</td></tr>
<tr><td colspan="5">interrupt</td></tr>
<tr><td colspan="5">description<br>(S)-=:PC,P PC:=($FFFE)</td></tr>
<tr><td colspan="5">c–fct<br>void cpu_6502_BRK_imp();</td></tr>
</table>

<table>
<tr><td>op<br>0x00</td><td>name<br>BRK</td><td>adr<br>imp</td><td colspan="2">NVBDIZC<br>--1-1--</td><td rowspan="4">logic<br>PC = PC + 1<br>bPoke(SP,PC.h)<br>SP = SP - 1<br>bPoke(SP,PC.l)<br>SP = SP - 1<br>bPoke(SP, (P|$10) )<br>SP = SP - 1<br>l = bPeek($FFFE)<br>h = bPeek($FFFF)<<8<br>PC = h|l</td></tr>
<tr><td colspan="3">example<br>BRK</td><td>bytes<br>1</td><td>cycles<br>0</td></tr>
<tr><td colspan="5">description<br>interrupt</td></tr>
<tr><td colspan="5">formal<br>(S)-=:PC,P PC:=($FFFE)</td></tr>
<tr><td colspan="5">c–fct<br>void cpu_6502_BRK_imp();</td><td>comment</td></tr>
</table>

<table>
<tr><td>op<br>0x01</td><td>name<br>ORA</td><td>adr<br>izx</td><td colspan="2">NVBDIZC<br>*----*-</td><td>logic<br>A = A | M<br>P.N = A.7<br>P.Z = (A==0) ? 1:0</td></tr>
<tr><td colspan="3">example<br>ORA ($A5,X)</td><td>bytes<br>2</td><td>cycles<br>6</td><td rowspan="4">comment</td></tr>
<tr><td colspan="5">description<br>or with accumulator</td></tr>
<tr><td colspan="5">formal<br>A:=A or adr</td></tr>
<tr><td colspan="5">c–fct<br>void cpu_6502_ORA_izx();</td></tr>
</table>

| op | name | adr | NVBDIZC |
|---|---|---|---|
| 0x05 | ORA | zp | *----*- |

| example | bytes | cycles |
|---|---|---|
| ORA $AB | 2 | 3 |

description
or with accumulator

formal
A:=A or adr

c–fct
```
void cpu_6502_ORA_zp();
```

logic
```
A = A | M
P.N = A.7
P.Z = (A==0) ? 1:0
```

comment

| op | name | adr | NVBDIZC |
|---|---|---|---|
| 0x06 | ASL | zp | *----** |

| example | bytes | cycles |
|---|---|---|
| ASL $AB | 2 | 5 |

description
arithmetic shift left

formal
adr:=adr*2

c–fct
```
void cpu_6502_ASL_zp();
```

logic
```
P.C = B.7
B = (B << 1) & $FE
P.N = B.7
P.Z = (B==0) ? 1:0
```

comment

| op | name | adr | NVBDIZC |
|---|---|---|---|
| 0x08 | PHP | imp | ------- |

| example | bytes | cycles |
|---|---|---|
| PHP | 1 | 3 |

description
push processor status (SR)

formal
(S)-:=P

c–fct
```
void cpu_6502_PHP_imp();
```

logic
```
bPoke(SP,P)
SP = SP - 1
```

comment

| op | name | adr | NVBDIZC |
|---|---|---|---|
| 0x09 | ORA | imm | *----*- |

| example | bytes | cycles |
|---|---|---|
| ORA #$AB | 2 | 2 |

description
or with accumulator

formal
A:=A or adr

c–fct
```
void cpu_6502_ORA_imm();
```

logic
```
A = A | M
P.N = A.7
P.Z = (A==0) ? 1:0
```

comment

| | | | | |
|---|---|---|---|---|
| op<br>0x0A | name<br>ASL | adr<br>imp | NVBDIZC<br>*----** | logic<br>P.C = B.7<br>B = (B << 1) & $FE<br>P.N = B.7<br>P.Z = (B==0) ? 1:0 |
| example<br>ASL | | bytes<br>1 | cycles<br>2 | comment |
| description<br>arithmetic shift left | | | | |
| formal<br>adr:=adr*2 | | | | |
| c–fct<br>void cpu_6502_ASL_imp(); | | | | |

| | | | | |
|---|---|---|---|---|
| op<br>0x0D | name<br>ORA | adr<br>abs | NVBDIZC<br>*----*- | logic<br>A = A \| M<br>P.N = A.7<br>P.Z = (A==0) ? 1:0 |
| example<br>ORA $ABCD | | bytes<br>3 | cycles<br>4 | comment |
| description<br>or with accumulator | | | | |
| formal<br>A:=A or adr | | | | |
| c–fct<br>void cpu_6502_ORA_abs(); | | | | |

| | | | | |
|---|---|---|---|---|
| op<br>0x0E | name<br>ASL | adr<br>abs | NVBDIZC<br>*----** | logic<br>P.C = B.7<br>B = (B << 1) & $FE<br>P.N = B.7<br>P.Z = (B==0) ? 1:0 |
| example<br>ASL $ABCD | | bytes<br>3 | cycles<br>6 | comment |
| description<br>arithmetic shift left | | | | |
| formal<br>adr:=adr*2 | | | | |
| c–fct<br>void cpu_6502_ASL_abs(); | | | | |

| | | | | |
|---|---|---|---|---|
| op<br>0x10 | name<br>BPL | adr<br>rel | NVBDIZC<br>------- | logic<br>if (P.N == 0) GOTO (PC+M) |
| example<br>BPL $AB | | bytes<br>2 | cycles<br>3 | comment |
| description<br>branch on plus (negative clear) | | | | |
| formal<br>branch on N=0 | | | | |
| c–fct<br>void cpu_6502_BPL_rel(); | | | | |

<table>
<tr><td>op<br>0x11</td><td>name<br>ORA</td><td>adr<br>izy</td><td colspan="2">NVBDIZC<br>*----*-</td><td>logic<br>A = A | M<br>P.N = A.7<br>P.Z = (A==0) ? 1:0</td></tr>
<tr><td colspan="3">example<br>ORA ($A5),X</td><td>bytes<br>2</td><td>cycles<br>5</td><td rowspan="4">comment</td></tr>
<tr><td colspan="5">description<br>or with accumulator</td></tr>
<tr><td colspan="5">formal<br>A:=A or adr</td></tr>
<tr><td colspan="5">c–fct<br>void cpu_6502_ORA_izy();</td></tr>
</table>

<table>
<tr><td>op<br>0x15</td><td>name<br>ORA</td><td>adr<br>zpx</td><td colspan="2">NVBDIZC<br>*----*-</td><td>logic<br>A = A | M<br>P.N = A.7<br>P.Z = (A==0) ? 1:0</td></tr>
<tr><td colspan="3">example<br>ORA $AB,X</td><td>bytes<br>2</td><td>cycles<br>4</td><td rowspan="4">comment</td></tr>
<tr><td colspan="5">description<br>or with accumulator</td></tr>
<tr><td colspan="5">formal<br>A:=A or adr</td></tr>
<tr><td colspan="5">c–fct<br>void cpu_6502_ORA_zpx();</td></tr>
</table>

<table>
<tr><td>op<br>0x16</td><td>name<br>ASL</td><td>adr<br>zpx</td><td colspan="2">NVBDIZC<br>*----**</td><td>logic<br>P.C = B.7<br>B = (B << 1) & $FE<br>P.N = B.7<br>P.Z = (B==0) ? 1:0</td></tr>
<tr><td colspan="3">example<br>ASL $AB,X</td><td>bytes<br>2</td><td>cycles<br>6</td><td rowspan="4">comment</td></tr>
<tr><td colspan="5">description<br>arithmetic shift left</td></tr>
<tr><td colspan="5">formal<br>adr:=adr*2</td></tr>
<tr><td colspan="5">c–fct<br>void cpu_6502_ASL_zpx();</td></tr>
</table>

<table>
<tr><td>op<br>0x18</td><td>name<br>CLC</td><td>adr<br>imp</td><td colspan="2">NVBDIZC<br>------0</td><td>logic<br>P.C = 0</td></tr>
<tr><td colspan="3">example<br>CLC</td><td>bytes<br>1</td><td>cycles<br>2</td><td rowspan="4">comment</td></tr>
<tr><td colspan="5">description<br>clear carry</td></tr>
<tr><td colspan="5">formal<br>C:=0</td></tr>
<tr><td colspan="5">c–fct<br>void cpu_6502_CLC_imp();</td></tr>
</table>

<table>
<tr><td>op<br>0x19</td><td>name<br>ORA</td><td>adr<br>aby</td><td colspan="2">NVBDIZC<br>*----*-</td><td>logic<br>A = A | M<br>P.N = A.7<br>P.Z = (A==0) ? 1:0</td></tr>
<tr><td colspan="3">example<br>ORA $ABCD,Y</td><td>bytes<br>3</td><td>cycles<br>4</td><td rowspan="4">comment</td></tr>
<tr><td colspan="5">description<br>or with accumulator</td></tr>
<tr><td colspan="5">formal<br>A:=A or adr</td></tr>
<tr><td colspan="5">c–fct<br>void cpu_6502_ORA_aby();</td></tr>
</table>

<table>
<tr><td>op<br>0x1D</td><td>name<br>ORA</td><td>adr<br>abx</td><td colspan="2">NVBDIZC<br>*----*-</td><td>logic<br>A = A | M<br>P.N = A.7<br>P.Z = (A==0) ? 1:0</td></tr>
<tr><td colspan="3">example<br>ORA $ABCD,X</td><td>bytes<br>3</td><td>cycles<br>4</td><td rowspan="4">comment</td></tr>
<tr><td colspan="5">description<br>or with accumulator</td></tr>
<tr><td colspan="5">formal<br>A:=A or adr</td></tr>
<tr><td colspan="5">c–fct<br>void cpu_6502_ORA_abx();</td></tr>
</table>

<table>
<tr><td>op<br>0x1E</td><td>name<br>ASL</td><td>adr<br>abx</td><td colspan="2">NVBDIZC<br>*----**</td><td>logic<br>P.C = B.7<br>B = (B << 1) & $FE<br>P.N = B.7<br>P.Z = (B==0) ? 1:0</td></tr>
<tr><td colspan="3">example<br>ASL $ABCD,X</td><td>bytes<br>3</td><td>cycles<br>7</td><td rowspan="4">comment</td></tr>
<tr><td colspan="5">description<br>arithmetic shift left</td></tr>
<tr><td colspan="5">formal<br>adr:=adr*2</td></tr>
<tr><td colspan="5">c–fct<br>void cpu_6502_ASL_abx();</td></tr>
</table>

<table>
<tr><td>op<br>0x20</td><td>name<br>JSR</td><td>adr<br>abs</td><td colspan="2">NVBDIZC<br>-------</td><td>logic<br>t = PC - 1<br>bPoke(SP,t.h)<br>SP = SP - 1<br>bPoke(SP,t.l)<br>SP = SP - 1<br>PC = $A5B6</td></tr>
<tr><td colspan="3">example<br>JSR $ABCD</td><td>bytes<br>3</td><td>cycles<br>6</td><td rowspan="4">comment</td></tr>
<tr><td colspan="5">description<br>jump subroutine</td></tr>
<tr><td colspan="5">formal<br>(S)-:=PC PC:=adr</td></tr>
<tr><td colspan="5">c–fct<br>void cpu_6502_JSR_abs();</td></tr>
</table>

| | | | | |
|---|---|---|---|---|
| op<br>0x21 | name<br>AND | adr<br>izx | NVBDIZC<br>`*----*-` | logic |
| example<br>AND ($A5,X) | | | bytes<br>2 | cycles<br>6 |
| description<br>and (with accumulator) | | | | |
| formal<br>A:=A&adr | | | | |
| c–fct<br>`void cpu_6502_AND_izx();` | | | | |
| comment | | | | |

```
A = A & M
P.N = A.7
P.Z = (A==0) ? 1:0
```

| | | | | |
|---|---|---|---|---|
| op<br>0x24 | name<br>BIT | adr<br>zp | NVBDIZC<br>`**---*-` | logic |
| example<br>BIT $AB | | | bytes<br>2 | cycles<br>3 |
| description<br>bit test | | | | |
| formal<br>N:=b7 V:=b6 Z:=A&adr | | | | |
| c–fct<br>`void cpu_6502_BIT_zp();` | | | | |
| comment | | | | |

```
t = A & M
P.N = t.7
P.V = t.6
P.Z = (t==0) ? 1:0
```

| | | | | |
|---|---|---|---|---|
| op<br>0x25 | name<br>AND | adr<br>zp | NVBDIZC<br>`*----*-` | logic |
| example<br>AND $AB | | | bytes<br>2 | cycles<br>3 |
| description<br>and (with accumulator) | | | | |
| formal<br>A:=A&adr | | | | |
| c–fct<br>`void cpu_6502_AND_zp();` | | | | |
| comment | | | | |

```
A = A & M
P.N = A.7
P.Z = (A==0) ? 1:0
```

| | | | | |
|---|---|---|---|---|
| op<br>0x26 | name<br>ROL | adr<br>zp | NVBDIZC<br>`*----**` | logic |
| example<br>ROL $AB | | | bytes<br>2 | cycles<br>5 |
| description<br>rotate left | | | | |
| formal<br>adr:=adr*2+C | | | | |
| c–fct<br>`void cpu_6502_ROL_zp();` | | | | |
| comment | | | | |

```
t = B.7
B = (B << 1) & $FE
B = B | P.C
P.C = t
P.Z = (B==0) ? 1:0
P.N = B.7
```

<table>
<tr><td>op<br>0x28</td><td>name<br>PLP</td><td>adr<br>imp</td><td colspan="2">NVBDIZC<br>**-****</td><td>logic<br>SP = SP + 1<br>P = bPeek(SP)</td></tr>
<tr><td colspan="3">example<br>PLP</td><td>bytes<br>1</td><td>cycles<br>4</td><td rowspan="4">comment</td></tr>
<tr><td colspan="5">description<br>pull processor status (SR)</td></tr>
<tr><td colspan="5">formal<br>P:=+(S)</td></tr>
<tr><td colspan="5">c–fct<br>void cpu_6502_PLP_imp();</td></tr>
</table>

<table>
<tr><td>op<br>0x29</td><td>name<br>AND</td><td>adr<br>imm</td><td colspan="2">NVBDIZC<br>*----*-</td><td>logic<br>A = A & M<br>P.N = A.7<br>P.Z = (A==0) ? 1:0</td></tr>
<tr><td colspan="3">example<br>AND #$AB</td><td>bytes<br>2</td><td>cycles<br>2</td><td rowspan="4">comment</td></tr>
<tr><td colspan="5">description<br>and (with accumulator)</td></tr>
<tr><td colspan="5">formal<br>A:=A&adr</td></tr>
<tr><td colspan="5">c–fct<br>void cpu_6502_AND_imm();</td></tr>
</table>

<table>
<tr><td>op<br>0x2A</td><td>name<br>ROL</td><td>adr<br>imp</td><td colspan="2">NVBDIZC<br>*----**</td><td>logic<br>t = B.7<br>B = (B << 1) & $FE<br>B = B | P.C<br>P.C = t<br>P.Z = (B==0) ? 1:0<br>P.N = B.7</td></tr>
<tr><td colspan="3">example<br>ROL</td><td>bytes<br>1</td><td>cycles<br>2</td><td rowspan="4">comment</td></tr>
<tr><td colspan="5">description<br>rotate left</td></tr>
<tr><td colspan="5">formal<br>adr:=adr*2+C</td></tr>
<tr><td colspan="5">c–fct<br>void cpu_6502_ROL_imp();</td></tr>
</table>

<table>
<tr><td>op<br>0x2C</td><td>name<br>BIT</td><td>adr<br>abs</td><td colspan="2">NVBDIZC<br>**---*-</td><td>logic<br>t = A & M<br>P.N = t.7<br>P.V = t.6<br>P.Z = (t==0) ? 1:0</td></tr>
<tr><td colspan="3">example<br>BIT $ABCD</td><td>bytes<br>3</td><td>cycles<br>4</td><td rowspan="4">comment</td></tr>
<tr><td colspan="5">description<br>bit test</td></tr>
<tr><td colspan="5">formal<br>N:=b7 V:=b6 Z:=A&adr</td></tr>
<tr><td colspan="5">c–fct<br>void cpu_6502_BIT_abs();</td></tr>
</table>

<table>
<tr><td>op<br>0x2D</td><td>name<br>AND</td><td>adr<br>abs</td><td colspan="2">NVBDIZC<br>*----*-</td><td>logic<br>A = A & M<br>P.N = A.7<br>P.Z = (A==0) ? 1:0</td></tr>
<tr><td colspan="3">example<br>AND $ABCD</td><td>bytes<br>3</td><td>cycles<br>4</td><td rowspan="4">comment</td></tr>
<tr><td colspan="5">description<br>and (with accumulator)</td></tr>
<tr><td colspan="5">formal<br>A:=A&adr</td></tr>
<tr><td colspan="5">c–fct<br>void cpu_6502_AND_abs();</td></tr>
</table>

<table>
<tr><td>op<br>0x2E</td><td>name<br>ROL</td><td>adr<br>abs</td><td colspan="2">NVBDIZC<br>*----**</td><td>logic<br>t = B.7<br>B = (B << 1) & $FE<br>B = B | P.C<br>P.C = t<br>P.Z = (B==0) ? 1:0<br>P.N = B.7</td></tr>
<tr><td colspan="3">example<br>ROL $ABCD</td><td>bytes<br>3</td><td>cycles<br>6</td><td rowspan="4">comment</td></tr>
<tr><td colspan="5">description<br>rotate left</td></tr>
<tr><td colspan="5">formal<br>adr:=adr*2+C</td></tr>
<tr><td colspan="5">c–fct<br>void cpu_6502_ROL_abs();</td></tr>
</table>

<table>
<tr><td>op<br>0x30</td><td>name<br>BMI</td><td>adr<br>rel</td><td colspan="2">NVBDIZC<br>-------</td><td>logic<br>if (P.N == 1) GOTO (PC+M)</td></tr>
<tr><td colspan="3">example<br>BMI $AB</td><td>bytes<br>2</td><td>cycles<br>2</td><td rowspan="4">comment</td></tr>
<tr><td colspan="5">description<br>branch on minus (negative set)</td></tr>
<tr><td colspan="5">formal<br>branch on N=1</td></tr>
<tr><td colspan="5">c–fct<br>void cpu_6502_BMI_rel();</td></tr>
</table>

<table>
<tr><td>op<br>0x31</td><td>name<br>AND</td><td>adr<br>izy</td><td colspan="2">NVBDIZC<br>*----*-</td><td>logic<br>A = A & M<br>P.N = A.7<br>P.Z = (A==0) ? 1:0</td></tr>
<tr><td colspan="3">example<br>AND ($A5),X</td><td>bytes<br>2</td><td>cycles<br>5</td><td rowspan="4">comment</td></tr>
<tr><td colspan="5">description<br>and (with accumulator)</td></tr>
<tr><td colspan="5">formal<br>A:=A&adr</td></tr>
<tr><td colspan="5">c–fct<br>void cpu_6502_AND_izy();</td></tr>
</table>

| | | | | |
|---|---|---|---|---|
| op<br>0x35 | name<br>AND | adr<br>zpx | NVBDIZC<br>*----*- | logic<br>A = A & M<br>P.N = A.7<br>P.Z = (A==0) ? 1:0 |
| example<br>AND $AB,X | | bytes<br>2 | cycles<br>4 | comment |
| description<br>and (with accumulator) | | | | |
| formal<br>A:=A&adr | | | | |
| c–fct<br>void cpu_6502_AND_zpx(); | | | | |

| | | | | |
|---|---|---|---|---|
| op<br>0x36 | name<br>ROL | adr<br>zpx | NVBDIZC<br>*-----** | logic<br>t = B.7<br>B = (B << 1) & $FE<br>B = B \| P.C<br>P.C = t<br>P.Z = (B==0) ? 1:0<br>P.N = B.7 |
| example<br>ROL $AB,X | | bytes<br>2 | cycles<br>6 | comment |
| description<br>rotate left | | | | |
| formal<br>adr:=adr*2+C | | | | |
| c–fct<br>void cpu_6502_ROL_zpx(); | | | | |

| | | | | |
|---|---|---|---|---|
| op<br>0x38 | name<br>SEC | adr<br>imp | NVBDIZC<br>------1 | logic<br>P.C = 1 |
| example<br>SEC | | bytes<br>1 | cycles<br>2 | comment |
| description<br>set carry | | | | |
| formal<br>C:=1 | | | | |
| c–fct<br>void cpu_6502_SEC_imp(); | | | | |

| | | | | |
|---|---|---|---|---|
| op<br>0x39 | name<br>AND | adr<br>aby | NVBDIZC<br>*----*- | logic<br>A = A & M<br>P.N = A.7<br>P.Z = (A==0) ? 1:0 |
| example<br>AND $ABCD,Y | | bytes<br>3 | cycles<br>4 | comment |
| description<br>and (with accumulator) | | | | |
| formal<br>A:=A&adr | | | | |
| c–fct<br>void cpu_6502_AND_aby(); | | | | |

<table>
<tr><td>op<br>0x3D</td><td>name<br>AND</td><td>adr<br>abx</td><td colspan="2">NVBDIZC<br>*----*-</td><td>logic<br>A = A & M<br>P.N = A.7<br>P.Z = (A==0) ? 1:0</td></tr>
<tr><td colspan="3">example<br>AND $ABCD,X</td><td>bytes<br>3</td><td>cycles<br>4</td><td rowspan="4">comment</td></tr>
<tr><td colspan="5">description<br>and (with accumulator)</td></tr>
<tr><td colspan="5">formal<br>A:=A&adr</td></tr>
<tr><td colspan="5">c–fct<br>void cpu_6502_AND_abx();</td></tr>
</table>

<table>
<tr><td>op<br>0x3E</td><td>name<br>ROL</td><td>adr<br>abx</td><td colspan="2">NVBDIZC<br>*-----**</td><td>logic<br>t = B.7<br>B = (B << 1) & $FE<br>B = B | P.C<br>P.C = t<br>P.Z = (B==0) ? 1:0<br>P.N = B.7</td></tr>
<tr><td colspan="3">example<br>ROL $ABCD,X</td><td>bytes<br>3</td><td>cycles<br>7</td><td rowspan="4">comment</td></tr>
<tr><td colspan="5">description<br>rotate left</td></tr>
<tr><td colspan="5">formal<br>adr:=adr*2+C</td></tr>
<tr><td colspan="5">c–fct<br>void cpu_6502_ROL_abx();</td></tr>
</table>

<table>
<tr><td>op<br>0x40</td><td>name<br>RTI</td><td>adr<br>imp</td><td colspan="2">NVBDIZC<br>**-****</td><td>logic<br>SP = SP - 1<br>P = bPeek(SP)<br>SP = SP - 1<br>l = bPeek(SP)<br>SP = SP - 1<br>h = bPeek(SP)<<8<br>PC = h|l</td></tr>
<tr><td colspan="3">example<br>RTI</td><td>bytes<br>1</td><td>cycles<br>6</td><td rowspan="4">comment</td></tr>
<tr><td colspan="5">description<br>return from interrupt</td></tr>
<tr><td colspan="5">formal<br>P,PC:=+(S)</td></tr>
<tr><td colspan="5">c–fct<br>void cpu_6502_RTI_imp();</td></tr>
</table>

<table>
<tr><td>op<br>0x41</td><td>name<br>EOR</td><td>adr<br>izx</td><td colspan="2">NVBDIZC<br>*----*-</td><td>logic<br>A = A ^ M<br>P.N = A.7<br>P.Z = (A==0) ? 1:0</td></tr>
<tr><td colspan="3">example<br>EOR ($A5,X)</td><td>bytes<br>2</td><td>cycles<br>6</td><td rowspan="4">comment</td></tr>
<tr><td colspan="5">description<br>exclusive or (with accumulator)</td></tr>
<tr><td colspan="5">formal<br>A:=A exor adr</td></tr>
<tr><td colspan="5">c–fct<br>void cpu_6502_EOR_izx();</td></tr>
</table>

| | | | | | |
|---|---|---|---|---|---|
| op<br>0x45 | name<br>EOR | adr<br>zp | NVBDIZC<br>*----*- | | logic<br>A = A ^ M<br>P.N = A.7<br>P.Z = (A==0) ? 1:0 |
| example<br>EOR $AB | | | bytes<br>2 | cycles<br>3 | comment |
| description<br>exclusive or (with accumulator) | | | | | |
| formal<br>A:=A exor adr | | | | | |
| c–fct<br>void cpu_6502_EOR_zp(); | | | | | |

| | | | | | |
|---|---|---|---|---|---|
| op<br>0x46 | name<br>LSR | adr<br>zp | NVBDIZC<br>*----** | | logic<br>P.N = 0<br>P.C = B.0<br>B = (B >> 1) & $7F<br>P.Z = (B==0) ? 1:0 |
| example<br>LSR $AB | | | bytes<br>2 | cycles<br>5 | comment |
| description<br>logical shift right | | | | | |
| formal<br>adr:=adr/2 | | | | | |
| c–fct<br>void cpu_6502_LSR_zp(); | | | | | |

| | | | | | |
|---|---|---|---|---|---|
| op<br>0x48 | name<br>PHA | adr<br>imp | NVBDIZC<br>------- | | logic<br>bPoke(SP,A)<br>SP = SP - 1 |
| example<br>PHA | | | bytes<br>1 | cycles<br>3 | comment |
| description<br>push accumulator | | | | | |
| formal<br>(S)-:=A | | | | | |
| c–fct<br>void cpu_6502_PHA_imp(); | | | | | |

| | | | | | |
|---|---|---|---|---|---|
| op<br>0x49 | name<br>EOR | adr<br>imm | NVBDIZC<br>*----*- | | logic<br>A = A ^ M<br>P.N = A.7<br>P.Z = (A==0) ? 1:0 |
| example<br>EOR #$AB | | | bytes<br>2 | cycles<br>2 | comment |
| description<br>exclusive or (with accumulator) | | | | | |
| formal<br>A:=A exor adr | | | | | |
| c–fct<br>void cpu_6502_EOR_imm(); | | | | | |

| | |
|---|---|
| op | 0x4A |
| name | LSR |
| adr | imp |
| NVBDIZC | *----** |
| example | LSR |
| bytes | 1 |
| cycles | 2 |
| description | logical shift right |
| formal | adr:=adr/2 |
| c–fct | void cpu_6502_LSR_imp(); |
| logic | P.N = 0<br>P.C = B.0<br>B = (B >> 1) & $7F<br>P.Z = (B==0) ? 1:0 |
| comment | |

| | |
|---|---|
| op | 0x4C |
| name | JMP |
| adr | abs |
| NVBDIZC | ------- |
| example | JMP $ABCD |
| bytes | 3 |
| cycles | 3 |
| description | jump |
| formal | PC:=adr |
| c–fct | void cpu_6502_JMP_abs(); |
| logic | PC = M |
| comment | |

| | |
|---|---|
| op | 0x4D |
| name | EOR |
| adr | abs |
| NVBDIZC | *----*- |
| example | EOR $ABCD |
| bytes | 3 |
| cycles | 4 |
| description | exclusive or (with accumulator) |
| formal | A:=A exor adr |
| c–fct | void cpu_6502_EOR_abs(); |
| logic | A = A ^ M<br>P.N = A.7<br>P.Z = (A==0) ? 1:0 |
| comment | |

| | |
|---|---|
| op | 0x4E |
| name | LSR |
| adr | abs |
| NVBDIZC | *----** |
| example | LSR $ABCD |
| bytes | 3 |
| cycles | 6 |
| description | logical shift right |
| formal | adr:=adr/2 |
| c–fct | void cpu_6502_LSR_abs(); |
| logic | P.N = 0<br>P.C = B.0<br>B = (B >> 1) & $7F<br>P.Z = (B==0) ? 1:0 |
| comment | |

|  |  |  |  |  |
|---|---|---|---|---|
| op<br>0x50 | name<br>BVC | adr<br>rel | NVBDIZC<br>------- | logic<br>`if (P.V == 0) GOTO (PC+M)` |
| example<br>BVC $AB |  | bytes<br>2 | cycles<br>3 | comment |
| description<br>branch on overflow clear |  |  |  |  |
| formal<br>branch on V=0 |  |  |  |  |
| c–fct<br>`void cpu_6502_BVC_rel();` |  |  |  |  |

|  |  |  |  |  |
|---|---|---|---|---|
| op<br>0x51 | name<br>EOR | adr<br>izy | NVBDIZC<br>*-----*- | logic<br>`A = A ^ M`<br>`P.N = A.7`<br>`P.Z = (A==0) ? 1:0` |
| example<br>EOR ($A5),X |  | bytes<br>2 | cycles<br>5 | comment |
| description<br>exclusive or (with accumulator) |  |  |  |  |
| formal<br>A:=A exor adr |  |  |  |  |
| c–fct<br>`void cpu_6502_EOR_izy();` |  |  |  |  |

|  |  |  |  |  |
|---|---|---|---|---|
| op<br>0x55 | name<br>EOR | adr<br>zpx | NVBDIZC<br>*-----*- | logic<br>`A = A ^ M`<br>`P.N = A.7`<br>`P.Z = (A==0) ? 1:0` |
| example<br>EOR $AB,X |  | bytes<br>2 | cycles<br>4 | comment |
| description<br>exclusive or (with accumulator) |  |  |  |  |
| formal<br>A:=A exor adr |  |  |  |  |
| c–fct<br>`void cpu_6502_EOR_zpx();` |  |  |  |  |

|  |  |  |  |  |
|---|---|---|---|---|
| op<br>0x56 | name<br>LSR | adr<br>zpx | NVBDIZC<br>*-----** | logic<br>`P.N = 0`<br>`P.C = B.0`<br>`B = (B >> 1) & $7F`<br>`P.Z = (B==0) ? 1:0` |
| example<br>LSR $AB,X |  | bytes<br>2 | cycles<br>6 | comment |
| description<br>logical shift right |  |  |  |  |
| formal<br>adr:=adr/2 |  |  |  |  |
| c–fct<br>`void cpu_6502_LSR_zpx();` |  |  |  |  |

| | | | | | |
|---|---|---|---|---|---|
| op<br>0x58 | name<br>CLI | adr<br>imp | NVBDIZC<br>----0-- | | logic<br>P.I = 0 |
| example<br>CLI | | | bytes<br>1 | cycles<br>2 | comment |
| description<br>clear interrupt disable | | | | | |
| formal<br>I:=0 | | | | | |
| c–fct<br>void cpu_6502_CLI_imp(); | | | | | |

| | | | | | |
|---|---|---|---|---|---|
| op<br>0x59 | name<br>EOR | adr<br>aby | NVBDIZC<br>*----*- | | logic<br>A = A ^ M<br>P.N = A.7<br>P.Z = (A==0) ? 1:0 |
| example<br>EOR $ABCD,Y | | | bytes<br>3 | cycles<br>4 | comment |
| description<br>exclusive or (with accumulator) | | | | | |
| formal<br>A:=A exor adr | | | | | |
| c–fct<br>void cpu_6502_EOR_aby(); | | | | | |

| | | | | | |
|---|---|---|---|---|---|
| op<br>0x5D | name<br>EOR | adr<br>abx | NVBDIZC<br>*----*- | | logic<br>A = A ^ M<br>P.N = A.7<br>P.Z = (A==0) ? 1:0 |
| example<br>EOR $ABCD,X | | | bytes<br>3 | cycles<br>4 | comment |
| description<br>exclusive or (with accumulator) | | | | | |
| formal<br>A:=A exor adr | | | | | |
| c–fct<br>void cpu_6502_EOR_abx(); | | | | | |

| | | | | | |
|---|---|---|---|---|---|
| op<br>0x5E | name<br>LSR | adr<br>abx | NVBDIZC<br>*----** | | logic<br>P.N = 0<br>P.C = B.0<br>B = (B >> 1) & $7F<br>P.Z = (B==0) ? 1:0 |
| example<br>LSR $ABCD,X | | | bytes<br>3 | cycles<br>7 | comment |
| description<br>logical shift right | | | | | |
| formal<br>adr:=adr/2 | | | | | |
| c–fct<br>void cpu_6502_LSR_abx(); | | | | | |

| op | name | adr | NVBDIZC |
|---|---|---|---|
| 0x60 | RTS | imp | ------- |

| example | bytes | cycles |
|---|---|---|
| RTS | 1 | 6 |

description
return from subroutine

formal
PC:=+(S)

c–fct
`void cpu_6502_RTS_imp();`

logic
```
SP = SP + 1
l = bPeek(SP)
SP = SP + 1
h = bPeek(SP)<<8
PC = (h|l) +1
```

comment

| op | name | adr | NVBDIZC |
|---|---|---|---|
| 0x61 | ADC | izx | **---** |

| example | bytes | cycles |
|---|---|---|
| ADC ($A5,X) | 2 | 6 |

description
add with carry

formal
A:=A+adr

c–fct
`void cpu_6502_ADC_izx();`

logic
```
t = A + M + P.C
P.V = (A.7!=t.7) ? 1:0
P.N = A.7
P.Z = (t==0) ? 1:0
IF (P.D)
  t = bcd(A) + bcd(M) + P.C
  P.C = (t>99) ? 1:0
ELSE
  P.C = (t>255) ? 1:0
A = t & 0xFF
```

comment

| op | name | adr | NVBDIZC |
|---|---|---|---|
| 0x65 | ADC | zp | **---** |

| example | bytes | cycles |
|---|---|---|
| ADC $AB | 2 | 3 |

description
add with carry

formal
A:=A+adr

c–fct
`void cpu_6502_ADC_zp();`

logic
```
t = A + M + P.C
P.V = (A.7!=t.7) ? 1:0
P.N = A.7
P.Z = (t==0) ? 1:0
IF (P.D)
  t = bcd(A) + bcd(M) + P.C
  P.C = (t>99) ? 1:0
ELSE
  P.C = (t>255) ? 1:0
A = t & 0xFF
```

comment

| op | name | adr | NVBDIZC |
|---|---|---|---|
| 0x66 | ROR | zp | *----** |

| example | bytes | cycles |
|---|---|---|
| ROR $AB | 2 | 5 |

description
rotate right

formal
adr:=adr/2+C*128

c–fct
`void cpu_6502_ROR_zp();`

logic
```
t = B.0
B = (B >> 1) & $7F
B = B | ((P.C) ? $80:$00)
P.C = t
P.Z = (B==0) ? 1:0
P.N = B.7
```

comment

| op | name | adr | NVBDIZC |
|---|---|---|---|
| 0x68 | PLA | imp | *----*- |

| example | bytes | cycles |
|---|---|---|
| PLA | 1 | 4 |

description
pull accumulator

formal
A:=+(S)

c–fct
`void cpu_6502_PLA_imp();`

logic
```
SP = SP + 1
A = bPeek(SP)
P.N = A.7
P.Z = (A==0) ? 1:0
```

comment

| op | name | adr | NVBDIZC |
|---|---|---|---|
| 0x69 | ADC | imm | **---** |

| example | bytes | cycles |
|---|---|---|
| ADC #$AB | 2 | 2 |

description
add with carry

formal
A:=A+adr

c–fct
`void cpu_6502_ADC_imm();`

logic
```
t = A + M + P.C
P.V = (A.7!=t.7) ? 1:0
P.N = A.7
P.Z = (t==0) ? 1:0
IF (P.D)
  t = bcd(A) + bcd(M) + P.C
  P.C = (t>99) ? 1:0
ELSE
  P.C = (t>255) ? 1:0
A = t & 0xFF
```

comment

| op | name | adr | NVBDIZC |
|---|---|---|---|
| 0x6A | ROR | imp | *----** |

| example | bytes | cycles |
|---|---|---|
| ROR | 1 | 2 |

description
rotate right

formal
adr:=adr/2+C*128

c–fct
`void cpu_6502_ROR_imp();`

logic
```
t = B.0
B = (B >> 1) & $7F
B = B | ((P.C) ? $80:$00)
P.C = t
P.Z = (B==0) ? 1:0
P.N = B.7
```

comment

| op | name | adr | NVBDIZC |
|---|---|---|---|
| 0x6C | JMP | ind | ------- |

| example | bytes | cycles |
|---|---|---|
| JMP $ABCD | 3 | 5 |

description
jump

formal
PC:=adr

c–fct
`void cpu_6502_JMP_ind();`

logic
```
PC = M
```

comment

| op | name | adr | NVBDIZC |
|---|---|---|---|
| 0x6D | ADC | abs | **---** |

| example | bytes | cycles |
|---|---|---|
| ADC $ABCD | 3 | 4 |

description
add with carry

formal
A:=A+adr

c–fct
`void cpu_6502_ADC_abs();`

logic

```
t = A + M + P.C
P.V = (A.7!=t.7) ? 1:0
P.N = A.7
P.Z = (t==0) ? 1:0
IF (P.D)
  t = bcd(A) + bcd(M) + P.C
  P.C = (t>99) ? 1:0
ELSE
  P.C = (t>255) ? 1:0
A = t & 0xFF
```

comment

| op | name | adr | NVBDIZC |
|---|---|---|---|
| 0x6E | ROR | abs | *----** |

| example | bytes | cycles |
|---|---|---|
| ROR $ABCD | 3 | 6 |

description
rotate right

formal
adr:=adr/2+C*128

c–fct
`void cpu_6502_ROR_abs();`

logic

```
t = B.0
B = (B >> 1) & $7F
B = B | ((P.C) ? $80:$00)
P.C = t
P.Z = (B==0) ? 1:0
P.N = B.7
```

comment

| op | name | adr | NVBDIZC |
|---|---|---|---|
| 0x70 | BVS | rel | ------- |

| example | bytes | cycles |
|---|---|---|
| BVS $AB | 2 | 2 |

description
branch on overflow set

formal
branch on V=1

c–fct
`void cpu_6502_BVS_rel();`

logic

```
if (P.V == 1) GOTO (PC+M)
```

comment

| op | name | adr | NVBDIZC |
|---|---|---|---|
| 0x71 | ADC | izy | **---** |

| example | bytes | cycles |
|---|---|---|
| ADC ($A5),X | 2 | 5 |

description
add with carry

formal
A:=A+adr

c–fct
`void cpu_6502_ADC_izy();`

logic

```
t = A + M + P.C
P.V = (A.7!=t.7) ? 1:0
P.N = A.7
P.Z = (t==0) ? 1:0
IF (P.D)
  t = bcd(A) + bcd(M) + P.C
  P.C = (t>99) ? 1:0
ELSE
  P.C = (t>255) ? 1:0
A = t & 0xFF
```

comment

| op | name | adr | NVBDIZC |
|---|---|---|---|
| 0x75 | ADC | zpx | **---** |

| example | bytes | cycles |
|---|---|---|
| ADC $AB,X | 2 | 4 |

| | |
|---|---|
| description | add with carry |
| formal | A:=A+adr |
| c–fct | void cpu_6502_ADC_zpx(); |

logic
```
t = A + M + P.C
P.V = (A.7!=t.7) ? 1:0
P.N = A.7
P.Z = (t==0) ? 1:0
IF (P.D)
  t = bcd(A) + bcd(M) + P.C
  P.C = (t>99) ? 1:0
ELSE
  P.C = (t>255) ? 1:0
A = t & 0xFF
```

comment

| op | name | adr | NVBDIZC |
|---|---|---|---|
| 0x76 | ROR | zpx | *-----** |

| example | bytes | cycles |
|---|---|---|
| ROR $AB,X | 2 | 6 |

| | |
|---|---|
| description | rotate right |
| formal | adr:=adr/2+C*128 |
| c–fct | void cpu_6502_ROR_zpx(); |

logic
```
t = B.0
B = (B >> 1) & $7F
B = B | ((P.C) ? $80:$00)
P.C = t
P.Z = (B==0) ? 1:0
P.N = B.7
```

comment

| op | name | adr | NVBDIZC |
|---|---|---|---|
| 0x78 | SEI | imp | ----1-- |

| example | bytes | cycles |
|---|---|---|
| SEI | 1 | 2 |

| | |
|---|---|
| description | set interrupt disable |
| formal | I:=1 |
| c–fct | void cpu_6502_SEI_imp(); |

logic
```
P.I = 1
```

comment

| op | name | adr | NVBDIZC |
|---|---|---|---|
| 0x79 | ADC | aby | **---** |

| example | bytes | cycles |
|---|---|---|
| ADC $ABCD,Y | 3 | 4 |

| | |
|---|---|
| description | add with carry |
| formal | A:=A+adr |
| c–fct | void cpu_6502_ADC_aby(); |

logic
```
t = A + M + P.C
P.V = (A.7!=t.7) ? 1:0
P.N = A.7
P.Z = (t==0) ? 1:0
IF (P.D)
  t = bcd(A) + bcd(M) + P.C
  P.C = (t>99) ? 1:0
ELSE
  P.C = (t>255) ? 1:0
A = t & 0xFF
```

comment

| op | name | adr | NVBDIZC |
|---|---|---|---|
| 0x7D | ADC | abx | **---** |

| example | bytes | cycles |
|---|---|---|
| ADC $ABCD,X | 3 | 4 |

description
add with carry

formal
A:=A+adr

c–fct
`void cpu_6502_ADC_abx();`

logic
```
t = A + M + P.C
P.V = (A.7!=t.7) ? 1:0
P.N = A.7
P.Z = (t==0) ? 1:0
IF (P.D)
  t = bcd(A) + bcd(M) + P.C
  P.C = (t>99) ? 1:0
ELSE
  P.C = (t>255) ? 1:0
A = t & 0xFF
```

comment

| op | name | adr | NVBDIZC |
|---|---|---|---|
| 0x7E | ROR | abx | *----** |

| example | bytes | cycles |
|---|---|---|
| ROR $ABCD,X | 3 | 7 |

description
rotate right

formal
adr:=adr/2+C*128

c–fct
`void cpu_6502_ROR_abx();`

logic
```
t = B.0
B = (B >> 1) & $7F
B = B | ((P.C) ? $80:$00)
P.C = t
P.Z = (B==0) ? 1:0
P.N = B.7
```

comment

| op | name | adr | NVBDIZC |
|---|---|---|---|
| 0x81 | STA | izx | ------- |

| example | bytes | cycles |
|---|---|---|
| STA ($A5,X) | 2 | 6 |

description
store accumulator

formal
adr:=A

c–fct
`void cpu_6502_STA_izx();`

logic
```
M = A
```

comment

| op | name | adr | NVBDIZC |
|---|---|---|---|
| 0x84 | STY | zp | ------- |

| example | bytes | cycles |
|---|---|---|
| STY $AB | 2 | 3 |

description
store Y

formal
adr:=Y

c–fct
`void cpu_6502_STY_zp();`

logic
```
M = Y
```

comment

<table>
<tr><td>op<br>0x85</td><td>name<br>STA</td><td>adr<br>zp</td><td colspan="2">NVBDIZC<br>-------</td><td>logic<br>M = A</td></tr>
<tr><td colspan="3">example<br>STA $AB</td><td>bytes<br>2</td><td>cycles<br>3</td><td rowspan="4">comment</td></tr>
<tr><td colspan="5">description<br>store accumulator</td></tr>
<tr><td colspan="5">formal<br>adr:=A</td></tr>
<tr><td colspan="5">c–fct<br>void cpu_6502_STA_zp();</td></tr>
</table>

<table>
<tr><td>op<br>0x86</td><td>name<br>STX</td><td>adr<br>zp</td><td colspan="2">NVBDIZC<br>-------</td><td>logic<br>M = X</td></tr>
<tr><td colspan="3">example<br>STX $AB</td><td>bytes<br>2</td><td>cycles<br>3</td><td rowspan="4">comment</td></tr>
<tr><td colspan="5">description<br>store X</td></tr>
<tr><td colspan="5">formal<br>adr:=X</td></tr>
<tr><td colspan="5">c–fct<br>void cpu_6502_STX_zp();</td></tr>
</table>

<table>
<tr><td>op<br>0x88</td><td>name<br>DEY</td><td>adr<br>imp</td><td colspan="2">NVBDIZC<br>*----*-</td><td>logic<br>Y = Y - 1<br>P.Z = (Y==0) ? 1:0<br>P.N = Y.7</td></tr>
<tr><td colspan="3">example<br>DEY</td><td>bytes<br>1</td><td>cycles<br>2</td><td rowspan="4">comment</td></tr>
<tr><td colspan="5">description<br>decrement Y</td></tr>
<tr><td colspan="5">formal<br>Y:=Y-1</td></tr>
<tr><td colspan="5">c–fct<br>void cpu_6502_DEY_imp();</td></tr>
</table>

<table>
<tr><td>op<br>0x8A</td><td>name<br>TXA</td><td>adr<br>imp</td><td colspan="2">NVBDIZC<br>*----*-</td><td>logic<br>A = X<br>P.N = A.7<br>P.Z = (A==0) ? 1:0</td></tr>
<tr><td colspan="3">example<br>TXA</td><td>bytes<br>1</td><td>cycles<br>2</td><td rowspan="4">comment</td></tr>
<tr><td colspan="5">description<br>transfer X to accumulator</td></tr>
<tr><td colspan="5">formal<br>A:=X</td></tr>
<tr><td colspan="5">c–fct<br>void cpu_6502_TXA_imp();</td></tr>
</table>

| | | | | | |
|---|---|---|---|---|---|
| op<br>0x8C | name<br>STY | adr<br>abs | NVBDIZC<br>------- | | logic `M = Y` |
| example<br>STY $ABCD | | | bytes<br>3 | cycles<br>4 | comment |
| description<br>store Y | | | | | |
| formal<br>adr:=Y | | | | | |
| c–fct<br>`void cpu_6502_STY_abs();` | | | | | |

| | | | | | |
|---|---|---|---|---|---|
| op<br>0x8D | name<br>STA | adr<br>abs | NVBDIZC<br>------- | | logic `M = A` |
| example<br>STA $ABCD | | | bytes<br>3 | cycles<br>4 | comment |
| description<br>store accumulator | | | | | |
| formal<br>adr:=A | | | | | |
| c–fct<br>`void cpu_6502_STA_abs();` | | | | | |

| | | | | | |
|---|---|---|---|---|---|
| op<br>0x8E | name<br>STX | adr<br>abs | NVBDIZC<br>------- | | logic `M = X` |
| example<br>STX $ABCD | | | bytes<br>3 | cycles<br>4 | comment |
| description<br>store X | | | | | |
| formal<br>adr:=X | | | | | |
| c–fct<br>`void cpu_6502_STX_abs();` | | | | | |

| | | | | | |
|---|---|---|---|---|---|
| op<br>0x90 | name<br>BCC | adr<br>rel | NVBDIZC<br>------- | | logic `if (P.C == 0) GOTO (PC+M)` |
| example<br>BCC $AB | | | bytes<br>2 | cycles<br>3 | comment |
| description<br>branch on carry clear | | | | | |
| formal<br>branch on C=0 | | | | | |
| c–fct<br>`void cpu_6502_BCC_rel();` | | | | | |

| | | | | | |
|---|---|---|---|---|---|
| op<br>0x91 | name<br>STA | adr<br>izy | NVBDIZC<br>------- | | logic M = A |
| example<br>STA ($A5),X | | | bytes<br>2 | cycles<br>6 | comment |
| description<br>store accumulator | | | | | |
| formal<br>adr:=A | | | | | |
| c–fct<br>void cpu_6502_STA_izy(); | | | | | |

| | | | | | |
|---|---|---|---|---|---|
| op<br>0x94 | name<br>STY | adr<br>zpx | NVBDIZC<br>------- | | logic M = Y |
| example<br>STY $AB,X | | | bytes<br>2 | cycles<br>4 | comment |
| description<br>store Y | | | | | |
| formal<br>adr:=Y | | | | | |
| c–fct<br>void cpu_6502_STY_zpx(); | | | | | |

| | | | | | |
|---|---|---|---|---|---|
| op<br>0x95 | name<br>STA | adr<br>zpx | NVBDIZC<br>------- | | logic M = A |
| example<br>STA $AB,X | | | bytes<br>2 | cycles<br>4 | comment |
| description<br>store accumulator | | | | | |
| formal<br>adr:=A | | | | | |
| c–fct<br>void cpu_6502_STA_zpx(); | | | | | |

| | | | | | |
|---|---|---|---|---|---|
| op<br>0x96 | name<br>STX | adr<br>zpy | NVBDIZC<br>------- | | logic M = X |
| example<br>STX $AB,X | | | bytes<br>2 | cycles<br>4 | comment |
| description<br>store X | | | | | |
| formal<br>adr:=X | | | | | |
| c–fct<br>void cpu_6502_STX_zpy(); | | | | | |

| | | | | | |
|---|---|---|---|---|---|
| op<br>0x98 | name<br>TYA | adr<br>imp | NVBDIZC<br>`*----*-` | | logic<br>comment |
| example<br>TYA | | | bytes<br>1 | cycles<br>2 | |
| description<br>transfer Y to accumulator | | | | | |
| formal<br>A:=Y | | | | | |
| c–fct<br>`void cpu_6502_TYA_imp();` | | | | | |

| | | | | | |
|---|---|---|---|---|---|
| op<br>0x99 | name<br>STA | adr<br>aby | NVBDIZC<br>`-------` | | logic `M = A` |
| example<br>STA $ABCD,Y | | | bytes<br>3 | cycles<br>5 | comment |
| description<br>store accumulator | | | | | |
| formal<br>adr:=A | | | | | |
| c–fct<br>`void cpu_6502_STA_aby();` | | | | | |

| | | | | | |
|---|---|---|---|---|---|
| op<br>0x9A | name<br>TXS | adr<br>imp | NVBDIZC<br>`-------` | | logic `SP = X` |
| example<br>TXS | | | bytes<br>1 | cycles<br>2 | comment |
| description<br>transfer X to stack pointer | | | | | |
| formal<br>S:=X | | | | | |
| c–fct<br>`void cpu_6502_TXS_imp();` | | | | | |

| | | | | | |
|---|---|---|---|---|---|
| op<br>0x9D | name<br>STA | adr<br>abx | NVBDIZC<br>`-------` | | logic `M = A` |
| example<br>STA $ABCD,X | | | bytes<br>3 | cycles<br>5 | comment |
| description<br>store accumulator | | | | | |
| formal<br>adr:=A | | | | | |
| c–fct<br>`void cpu_6502_STA_abx();` | | | | | |

| | | | | |
|---|---|---|---|---|
| op<br>0xA0 | name<br>LDY | adr<br>imm | NVBDIZC<br>`*----*-` | logic<br>`Y = M`<br>`P.N = Y.7`<br>`P.Z = (Y==0) ? 1:0` |
| example<br>LDY #$AB | | bytes<br>2 | cycles<br>2 | comment |
| description<br>load Y | | | | |
| formal<br>Y:=adr | | | | |
| c–fct<br>`void cpu_6502_LDY_imm();` | | | | |

| | | | | |
|---|---|---|---|---|
| op<br>0xA1 | name<br>LDA | adr<br>izx | NVBDIZC<br>`*----*-` | logic<br>`A = M`<br>`P.N = A.7`<br>`P.Z = (A==0) ? 1:0` |
| example<br>LDA ($A5,X) | | bytes<br>2 | cycles<br>6 | comment |
| description<br>load accumulator | | | | |
| formal<br>A:=adr | | | | |
| c–fct<br>`void cpu_6502_LDA_izx();` | | | | |

| | | | | |
|---|---|---|---|---|
| op<br>0xA2 | name<br>LDX | adr<br>imm | NVBDIZC<br>`*----*-` | logic<br>`X = M`<br>`P.N = X.7`<br>`P.Z = (X==0) ? 1:0` |
| example<br>LDX #$AB | | bytes<br>2 | cycles<br>2 | comment |
| description | | | | |
| formal<br>X:=adr | | | | |
| c–fct<br>`void cpu_6502_LDX_imm();` | | | | |

| | | | | |
|---|---|---|---|---|
| op<br>0xA4 | name<br>LDY | adr<br>zp | NVBDIZC<br>`*----*-` | logic<br>`Y = M`<br>`P.N = Y.7`<br>`P.Z = (Y==0) ? 1:0` |
| example<br>LDY $AB | | bytes<br>2 | cycles<br>3 | comment |
| description<br>load Y | | | | |
| formal<br>Y:=adr | | | | |
| c–fct<br>`void cpu_6502_LDY_zp();` | | | | |

| | | | | | |
|---|---|---|---|---|---|
| op<br>0xA5 | name<br>LDA | adr<br>zp | NVBDIZC<br>*----*- | | logic<br>A = M<br>P.N = A.7<br>P.Z = (A==0) ? 1:0 |
| example<br>LDA $AB | | | bytes<br>2 | cycles<br>3 | comment |
| description<br>load accumulator | | | | | |
| formal<br>A:=adr | | | | | |
| c–fct<br>void cpu_6502_LDA_zp(); | | | | | |

| | | | | | |
|---|---|---|---|---|---|
| op<br>0xA6 | name<br>LDX | adr<br>zp | NVBDIZC<br>*----*- | | logic<br>X = M<br>P.N = X.7<br>P.Z = (X==0) ? 1:0 |
| example<br>LDX $AB | | | bytes<br>2 | cycles<br>3 | comment |
| description | | | | | |
| formal<br>X:=adr | | | | | |
| c–fct<br>void cpu_6502_LDX_zp(); | | | | | |

| | | | | | |
|---|---|---|---|---|---|
| op<br>0xA8 | name<br>TAY | adr<br>imp | NVBDIZC<br>*----*- | | logic<br>Y = A<br>P.N = Y.7<br>P.Z = (Y==0) ? 1:0 |
| example<br>TAY | | | bytes<br>1 | cycles<br>2 | comment |
| description<br>transfer accumulator to Y | | | | | |
| formal<br>Y:=A | | | | | |
| c–fct<br>void cpu_6502_TAY_imp(); | | | | | |

| | | | | | |
|---|---|---|---|---|---|
| op<br>0xA9 | name<br>LDA | adr<br>imm | NVBDIZC<br>*----*- | | logic<br>A = M<br>P.N = A.7<br>P.Z = (A==0) ? 1:0 |
| example<br>LDA #$AB | | | bytes<br>2 | cycles<br>2 | comment |
| description<br>load accumulator | | | | | |
| formal<br>A:=adr | | | | | |
| c–fct<br>void cpu_6502_LDA_imm(); | | | | | |

| | | | |
|---|---|---|---|
| op<br>0xAA | name<br>TAX | adr<br>imp | NVBDIZC<br>*----*- |
| example<br>TAX | | bytes<br>1 | cycles<br>2 |
| description<br>transfer accumulator to X | | | |
| formal<br>X:=A | | | |
| c–fct<br>void cpu_6502_TAX_imp(); | | | |

logic
```
X = A
P.N = X.7
P.Z = (X==0) ? 1:0
```

comment

| | | | |
|---|---|---|---|
| op<br>0xAC | name<br>LDY | adr<br>abs | NVBDIZC<br>*----*- |
| example<br>LDY $ABCD | | bytes<br>3 | cycles<br>4 |
| description<br>load Y | | | |
| formal<br>Y:=adr | | | |
| c–fct<br>void cpu_6502_LDY_abs(); | | | |

logic
```
Y = M
P.N = Y.7
P.Z = (Y==0) ? 1:0
```

comment

| | | | |
|---|---|---|---|
| op<br>0xAD | name<br>LDA | adr<br>abs | NVBDIZC<br>*----*- |
| example<br>LDA $ABCD | | bytes<br>3 | cycles<br>4 |
| description<br>load accumulator | | | |
| formal<br>A:=adr | | | |
| c–fct<br>void cpu_6502_LDA_abs(); | | | |

logic
```
A = M
P.N = A.7
P.Z = (A==0) ? 1:0
```

comment

| | | | |
|---|---|---|---|
| op<br>0xAE | name<br>LDX | adr<br>abs | NVBDIZC<br>*----*- |
| example<br>LDX $ABCD | | bytes<br>3 | cycles<br>4 |
| description | | | |
| formal<br>X:=adr | | | |
| c–fct<br>void cpu_6502_LDX_abs(); | | | |

logic
```
X = M
P.N = X.7
P.Z = (X==0) ? 1:0
```

comment

| | | | | | |
|---|---|---|---|---|---|
| op<br>0xB0 | name<br>BCS | adr<br>rel | NVBDIZC<br>------- | | logic<br>if (P.C == 1) GOTO (PC+M) |
| example<br>BCS $AB | | | bytes<br>2 | cycles<br>2 | comment |
| description<br>branch on carry set | | | | | |
| formal<br>branch on C=1 | | | | | |
| c–fct<br>void cpu_6502_BCS_rel(); | | | | | |

| | | | | | |
|---|---|---|---|---|---|
| op<br>0xB1 | name<br>LDA | adr<br>izy | NVBDIZC<br>*----*- | | logic<br>A = M<br>P.N = A.7<br>P.Z = (A==0) ? 1:0 |
| example<br>LDA ($A5),X | | | bytes<br>2 | cycles<br>5 | comment |
| description<br>load accumulator | | | | | |
| formal<br>A:=adr | | | | | |
| c–fct<br>void cpu_6502_LDA_izy(); | | | | | |

| | | | | | |
|---|---|---|---|---|---|
| op<br>0xB4 | name<br>LDY | adr<br>zpx | NVBDIZC<br>*----*- | | logic<br>Y = M<br>P.N = Y.7<br>P.Z = (Y==0) ? 1:0 |
| example<br>LDY $AB,X | | | bytes<br>2 | cycles<br>4 | comment |
| description<br>load Y | | | | | |
| formal<br>Y:=adr | | | | | |
| c–fct<br>void cpu_6502_LDY_zpx(); | | | | | |

| | | | | | |
|---|---|---|---|---|---|
| op<br>0xB5 | name<br>LDA | adr<br>zpx | NVBDIZC<br>*----*- | | logic<br>A = M<br>P.N = A.7<br>P.Z = (A==0) ? 1:0 |
| example<br>LDA $AB,X | | | bytes<br>2 | cycles<br>4 | comment |
| description<br>load accumulator | | | | | |
| formal<br>A:=adr | | | | | |
| c–fct<br>void cpu_6502_LDA_zpx(); | | | | | |

| | |
|---|---|
| op | 0xB6 |
| name | LDX |
| adr | zpy |
| NVBDIZC | *-----*- |
| example | LDX $AB,X |
| bytes | 2 |
| cycles | 4 |
| description | |
| formal | X:=adr |
| c–fct | `void cpu_6502_LDX_zpy();` |
| logic | `X = M`<br>`P.N = X.7`<br>`P.Z = (X==0) ? 1:0` |
| comment | |

| | |
|---|---|
| op | 0xB8 |
| name | CLV |
| adr | imp |
| NVBDIZC | -0------ |
| example | CLV |
| bytes | 1 |
| cycles | 2 |
| description | clear overflow |
| formal | V:=0 |
| c–fct | `void cpu_6502_CLV_imp();` |
| logic | `P.V = 0` |
| comment | |

| | |
|---|---|
| op | 0xB9 |
| name | LDA |
| adr | aby |
| NVBDIZC | *-----*- |
| example | LDA $ABCD,Y |
| bytes | 3 |
| cycles | 4 |
| description | load accumulator |
| formal | A:=adr |
| c–fct | `void cpu_6502_LDA_aby();` |
| logic | `A = M`<br>`P.N = A.7`<br>`P.Z = (A==0) ? 1:0` |
| comment | |

| | |
|---|---|
| op | 0xBA |
| name | TSX |
| adr | imp |
| NVBDIZC | *-----*- |
| example | TSX |
| bytes | 1 |
| cycles | 2 |
| description | transfer stack pointer to X |
| formal | X:=S |
| c–fct | `void cpu_6502_TSX_imp();` |
| logic | `X = SP`<br>`P.N = X.7`<br>`P.Z = (X==0) ? 1:0` |
| comment | |

| | | | | |
|---|---|---|---|---|
| op | name | adr | NVBDIZC | logic |
| 0xBC | LDY | abx | *----*- | Y = M<br>P.N = Y.7<br>P.Z = (Y==0) ? 1:0 |
| example | bytes | cycles | | comment |
| LDY $ABCD,X | 3 | 4 | | |
| description | | | | |
| load Y | | | | |
| formal | | | | |
| Y:=adr | | | | |
| c–fct | | | | |
| void cpu_6502_LDY_abx(); | | | | |

| | | | | |
|---|---|---|---|---|
| op | name | adr | NVBDIZC | logic |
| 0xBD | LDA | abx | *----*- | A = M<br>P.N = A.7<br>P.Z = (A==0) ? 1:0 |
| example | bytes | cycles | | comment |
| LDA $ABCD,X | 3 | 4 | | |
| description | | | | |
| load accumulator | | | | |
| formal | | | | |
| A:=adr | | | | |
| c–fct | | | | |
| void cpu_6502_LDA_abx(); | | | | |

| | | | | |
|---|---|---|---|---|
| op | name | adr | NVBDIZC | logic |
| 0xBE | LDX | aby | *----*- | X = M<br>P.N = X.7<br>P.Z = (X==0) ? 1:0 |
| example | bytes | cycles | | comment |
| LDX $ABCD,Y | 3 | 4 | | |
| description | | | | |
| | | | | |
| formal | | | | |
| X:=adr | | | | |
| c–fct | | | | |
| void cpu_6502_LDX_aby(); | | | | |

| | | | | |
|---|---|---|---|---|
| op | name | adr | NVBDIZC | logic |
| 0xC0 | CPY | imm | *----** | t = Y - M<br>P.N = t.7<br>P.C = (Y>=M) ? 1:0<br>P.Z = (t==0) ? 1:0 |
| example | bytes | cycles | | comment |
| CPY #$AB | 2 | 2 | | |
| description | | | | |
| compare with Y | | | | |
| formal | | | | |
| Y-adr | | | | |
| c–fct | | | | |
| void cpu_6502_CPY_imm(); | | | | |

| | | | | |
|---|---|---|---|---|
| op<br>0xC1 | name<br>CMP | adr<br>izx | NVBDIZC<br>*----** | |
| example<br>CMP ($A5,X) | | | bytes<br>2 | cycles<br>6 |
| description<br>compare (with accumulator) | | | | |
| formal<br>A-adr | | | | |
| c–fct<br>void cpu_6502_CMP_izx(); | | | | |

logic

```
t = A - M
P.N = t.7
P.C = (A>=M) ? 1:0
P.Z = (t==0) ? 1:0
```

comment

| | | | | |
|---|---|---|---|---|
| op<br>0xC4 | name<br>CPY | adr<br>zp | NVBDIZC<br>*----** | |
| example<br>CPY $AB | | | bytes<br>2 | cycles<br>3 |
| description<br>compare with Y | | | | |
| formal<br>Y-adr | | | | |
| c–fct<br>void cpu_6502_CPY_zp(); | | | | |

logic

```
t = Y - M
P.N = t.7
P.C = (Y>=M) ? 1:0
P.Z = (t==0) ? 1:0
```

comment

| | | | | |
|---|---|---|---|---|
| op<br>0xC5 | name<br>CMP | adr<br>zp | NVBDIZC<br>*----** | |
| example<br>CMP $AB | | | bytes<br>2 | cycles<br>3 |
| description<br>compare (with accumulator) | | | | |
| formal<br>A-adr | | | | |
| c–fct<br>void cpu_6502_CMP_zp(); | | | | |

logic

```
t = A - M
P.N = t.7
P.C = (A>=M) ? 1:0
P.Z = (t==0) ? 1:0
```

comment

| | | | | |
|---|---|---|---|---|
| op<br>0xC6 | name<br>DEC | adr<br>zp | NVBDIZC<br>*----*- | |
| example<br>DEC $AB | | | bytes<br>2 | cycles<br>5 |
| description<br>decrement | | | | |
| formal<br>adr:=adr-1 | | | | |
| c–fct<br>void cpu_6502_DEC_zp(); | | | | |

logic

```
M = (M - 1) & $FF
P.N = M.7
P.Z = (M==0) ? 1:0
```

comment

| | | | | |
|---|---|---|---|---|
| op<br>0xC8 | name<br>INY | adr<br>imp | NVBDIZC<br>*----*- | logic<br>Y = Y + 1<br>P.Z = (Y==0) ? 1:0<br>P.N = Y.7 |
| example<br>INY | | bytes<br>1 | cycles<br>2 | comment |
| description<br>increment Y | | | | |
| formal<br>Y:=Y+1 | | | | |
| c–fct<br>void cpu_6502_INY_imp(); | | | | |

| | | | | |
|---|---|---|---|---|
| op<br>0xC9 | name<br>CMP | adr<br>imm | NVBDIZC<br>*----** | logic<br>t = A - M<br>P.N = t.7<br>P.C = (A>=M) ? 1:0<br>P.Z = (t==0) ? 1:0 |
| example<br>CMP #$AB | | bytes<br>2 | cycles<br>2 | comment |
| description<br>compare (with accumulator) | | | | |
| formal<br>A-adr | | | | |
| c–fct<br>void cpu_6502_CMP_imm(); | | | | |

| | | | | |
|---|---|---|---|---|
| op<br>0xCA | name<br>DEX | adr<br>imp | NVBDIZC<br>*----*- | logic<br>X = X - 1<br>P.Z = (X==0) ? 1:0<br>P.N = X.7 |
| example<br>DEX | | bytes<br>1 | cycles<br>2 | comment |
| description<br>decrement X | | | | |
| formal<br>X:=X-1 | | | | |
| c–fct<br>void cpu_6502_DEX_imp(); | | | | |

| | | | | |
|---|---|---|---|---|
| op<br>0xCC | name<br>CPY | adr<br>abs | NVBDIZC<br>*----** | logic<br>t = Y - M<br>P.N = t.7<br>P.C = (Y>=M) ? 1:0<br>P.Z = (t==0) ? 1:0 |
| example<br>CPY $ABCD | | bytes<br>3 | cycles<br>4 | comment |
| description<br>compare with Y | | | | |
| formal<br>Y-adr | | | | |
| c–fct<br>void cpu_6502_CPY_abs(); | | | | |

| | | | | | |
|---|---|---|---|---|---|
| op | name | adr | NVBDIZC | | logic |
| 0xCD | CMP | abs | `*----**` | | `t = A - M`<br>`P.N = t.7`<br>`P.C = (A>=M) ? 1:0`<br>`P.Z = (t==0) ? 1:0` |
| example | | | bytes | cycles | comment |
| CMP $ABCD | | | 3 | 4 | |
| description | | | | | |
| compare (with accumulator) | | | | | |
| formal | | | | | |
| A-adr | | | | | |
| c–fct | | | | | |
| `void cpu_6502_CMP_abs();` | | | | | |

| | | | | | |
|---|---|---|---|---|---|
| op | name | adr | NVBDIZC | | logic |
| 0xCE | DEC | abs | `*-----*-` | | `M = (M - 1) & $FF`<br>`P.N = M.7`<br>`P.Z = (M==0) ? 1:0` |
| example | | | bytes | cycles | comment |
| DEC $ABCD | | | 3 | 6 | |
| description | | | | | |
| decrement | | | | | |
| formal | | | | | |
| adr:=adr-1 | | | | | |
| c–fct | | | | | |
| `void cpu_6502_DEC_abs();` | | | | | |

| | | | | | |
|---|---|---|---|---|---|
| op | name | adr | NVBDIZC | | logic |
| 0xD0 | BNE | rel | `-------` | | `if (P.Z == 0) GOTO (PC+M)` |
| example | | | bytes | cycles | comment |
| BNE $AB | | | 2 | 3 | |
| description | | | | | |
| branch on not equal (zero clear) | | | | | |
| formal | | | | | |
| branch on Z=0 | | | | | |
| c–fct | | | | | |
| `void cpu_6502_BNE_rel();` | | | | | |

| | | | | | |
|---|---|---|---|---|---|
| op | name | adr | NVBDIZC | | logic |
| 0xD1 | CMP | izy | `*----**` | | `t = A - M`<br>`P.N = t.7`<br>`P.C = (A>=M) ? 1:0`<br>`P.Z = (t==0) ? 1:0` |
| example | | | bytes | cycles | comment |
| CMP ($A5),X | | | 2 | 5 | |
| description | | | | | |
| compare (with accumulator) | | | | | |
| formal | | | | | |
| A-adr | | | | | |
| c–fct | | | | | |
| `void cpu_6502_CMP_izy();` | | | | | |

| | | | | |
|---|---|---|---|---|
| op<br>0xD5 | name<br>CMP | adr<br>zpx | NVBDIZC<br>*----** | |
| example<br>CMP $AB,X | | | bytes<br>2 | cycles<br>4 |
| description<br>compare (with accumulator) | | | | |
| formal<br>A-adr | | | | |
| c–fct<br>void cpu_6502_CMP_zpx(); | | | | |

logic
```
t = A - M
P.N = t.7
P.C = (A>=M) ? 1:0
P.Z = (t==0) ? 1:0
```

comment

| | | | | |
|---|---|---|---|---|
| op<br>0xD6 | name<br>DEC | adr<br>zpx | NVBDIZC<br>*----*- | |
| example<br>DEC $AB,X | | | bytes<br>2 | cycles<br>6 |
| description<br>decrement | | | | |
| formal<br>adr:=adr-1 | | | | |
| c–fct<br>void cpu_6502_DEC_zpx(); | | | | |

logic
```
M = (M - 1) & $FF
P.N = M.7
P.Z = (M==0) ? 1:0
```

comment

| | | | | |
|---|---|---|---|---|
| op<br>0xD8 | name<br>CLD | adr<br>imp | NVBDIZC<br>---0--- | |
| example<br>CLD | | | bytes<br>1 | cycles<br>2 |
| description<br>clear decimal | | | | |
| formal<br>D:=0 | | | | |
| c–fct<br>void cpu_6502_CLD_imp(); | | | | |

logic
```
P.D = 0
```

comment

| | | | | |
|---|---|---|---|---|
| op<br>0xD9 | name<br>CMP | adr<br>aby | NVBDIZC<br>*----** | |
| example<br>CMP $ABCD,Y | | | bytes<br>3 | cycles<br>4 |
| description<br>compare (with accumulator) | | | | |
| formal<br>A-adr | | | | |
| c–fct<br>void cpu_6502_CMP_aby(); | | | | |

logic
```
t = A - M
P.N = t.7
P.C = (A>=M) ? 1:0
P.Z = (t==0) ? 1:0
```

comment

| op | name | adr | NVBDIZC |
|---|---|---|---|
| 0xDD | CMP | abx | *----** |

| example | bytes | cycles |
|---|---|---|
| CMP $ABCD,X | 3 | 4 |

description
compare (with accumulator)

formal
A-adr

c–fct
`void cpu_6502_CMP_abx();`

logic
```
t = A - M
P.N = t.7
P.C = (A>=M) ? 1:0
P.Z = (t==0) ? 1:0
```

comment

| op | name | adr | NVBDIZC |
|---|---|---|---|
| 0xDE | DEC | abx | *----*- |

| example | bytes | cycles |
|---|---|---|
| DEC $ABCD,X | 3 | 7 |

description
decrement

formal
adr:=adr-1

c–fct
`void cpu_6502_DEC_abx();`

logic
```
M = (M - 1) & $FF
P.N = M.7
P.Z = (M==0) ? 1:0
```

comment

| op | name | adr | NVBDIZC |
|---|---|---|---|
| 0xE0 | CPX | imm | *----** |

| example | bytes | cycles |
|---|---|---|
| CPX #$AB | 2 | 2 |

description
compare with X

formal
X-adr

c–fct
`void cpu_6502_CPX_imm();`

logic
```
t = X - M
P.N = t.7
P.C = (X>=M) ? 1:0
P.Z = (t==0) ? 1:0
```

comment

| op | name | adr | NVBDIZC |
|---|---|---|---|
| 0xE1 | SBC | izx | **---** |

| example | bytes | cycles |
|---|---|---|
| SBC ($A5,X) | 2 | 6 |

description
subtract with carry

formal
A:=A-adr

c–fct
`void cpu_6502_SBC_izx();`

logic
```
IF (P.D)
  t = bcd(A) - bcd(M) - !P.C
  P.V = (t>99 OR t<0) ? 1:0
ELSE
  t = A - M - !P.C
  P.V = (t>127 OR t<-128) ? 1:0
P.C = (t>=0) ? 1:0
P.N = t.7
P.Z = (t==0) ? 1:0
A = t & 0xFF
```

comment

| | | | |
|---|---|---|---|
| op<br>0xE4 | name<br>CPX | adr<br>zp | NVBDIZC<br>*----** |
| example<br>CPX $AB | | bytes<br>2 | cycles<br>3 |
| description<br>compare with X | | | |
| formal<br>X-adr | | | |
| c–fct<br>`void cpu_6502_CPX_zp();` | | | |

logic
```
t = X - M
P.N = t.7
P.C = (X>=M) ? 1:0
P.Z = (t==0) ? 1:0
```

comment

| | | | |
|---|---|---|---|
| op<br>0xE5 | name<br>SBC | adr<br>zp | NVBDIZC<br>**---** |
| example<br>SBC $AB | | bytes<br>2 | cycles<br>3 |
| description<br>subtract with carry | | | |
| formal<br>A:=A-adr | | | |
| c–fct<br>`void cpu_6502_SBC_zp();` | | | |

logic
```
IF (P.D)
  t = bcd(A) - bcd(M) - !P.C
  P.V = (t>99 OR t<0) ? 1:0
ELSE
  t = A - M - !P.C
  P.V = (t>127 OR t<-128) ? 1:0
P.C = (t>=0) ? 1:0
P.N = t.7
P.Z = (t==0) ? 1:0
A = t & 0xFF
```

comment

| | | | |
|---|---|---|---|
| op<br>0xE6 | name<br>INC | adr<br>zp | NVBDIZC<br>*----*- |
| example<br>INC $AB | | bytes<br>2 | cycles<br>5 |
| description<br>increment | | | |
| formal<br>adr:=adr+1 | | | |
| c–fct<br>`void cpu_6502_INC_zp();` | | | |

logic
```
M = (M + 1) & $FF
P.N = M.7
P.Z = (M==0) ? 1:0
```

comment

| | | | |
|---|---|---|---|
| op<br>0xE8 | name<br>INX | adr<br>imp | NVBDIZC<br>*----*- |
| example<br>INX | | bytes<br>1 | cycles<br>2 |
| description<br>increment X | | | |
| formal<br>X:=X+1 | | | |
| c–fct<br>`void cpu_6502_INX_imp();` | | | |

logic
```
X = X + 1
P.Z = (X==0) ? 1:0
P.N = X.7
```

comment

| | | | | |
|---|---|---|---|---|
| op<br>0xE9 | name<br>SBC | adr<br>imm | NVBDIZC<br>**---** | |
| example<br>SBC #$AB | | | bytes<br>2 | cycles<br>2 |
| description<br>subtract with carry | | | | |
| formal<br>A:=A-adr | | | | |
| c–fct<br>void cpu_6502_SBC_imm(); | | | | |

logic

```
IF (P.D)
  t = bcd(A) - bcd(M) - !P.C
  P.V = (t>99 OR t<0) ? 1:0
ELSE
  t = A - M - !P.C
  P.V = (t>127 OR t<-128) ? 1:0
P.C = (t>=0) ? 1:0
P.N = t.7
P.Z = (t==0) ? 1:0
A = t & 0xFF
```

comment

| | | | | |
|---|---|---|---|---|
| op<br>0xEA | name<br>NOP | adr<br>imp | NVBDIZC<br>------- | |
| example<br>NOP | | | bytes<br>1 | cycles<br>2 |
| description<br>no operation | | | | |
| formal | | | | |
| c–fct<br>void cpu_6502_NOP_imp(); | | | | |

logic

```
~none~
```

comment

| | | | | |
|---|---|---|---|---|
| op<br>0xEC | name<br>CPX | adr<br>abs | NVBDIZC<br>*----** | |
| example<br>CPX $ABCD | | | bytes<br>3 | cycles<br>4 |
| description<br>compare with X | | | | |
| formal<br>X-adr | | | | |
| c–fct<br>void cpu_6502_CPX_abs(); | | | | |

logic

```
t = X - M
P.N = t.7
P.C = (X>=M) ? 1:0
P.Z = (t==0) ? 1:0
```

comment

| | | | | |
|---|---|---|---|---|
| op<br>0xED | name<br>SBC | adr<br>abs | NVBDIZC<br>**---** | |
| example<br>SBC $ABCD | | | bytes<br>3 | cycles<br>4 |
| description<br>subtract with carry | | | | |
| formal<br>A:=A-adr | | | | |
| c–fct<br>void cpu_6502_SBC_abs(); | | | | |

logic

```
IF (P.D)
  t = bcd(A) - bcd(M) - !P.C
  P.V = (t>99 OR t<0) ? 1:0
ELSE
  t = A - M - !P.C
  P.V = (t>127 OR t<-128) ? 1:0
P.C = (t>=0) ? 1:0
P.N = t.7
P.Z = (t==0) ? 1:0
A = t & 0xFF
```

comment

| | | | | | |
|---|---|---|---|---|---|
| op<br>0xEE | name<br>INC | adr<br>abs | NVBDIZC<br>*----*- | | |
| example<br>INC $ABCD | | | | bytes<br>3 | cycles<br>6 |
| description<br>increment | | | | | |
| formal<br>adr:=adr+1 | | | | | |
| c–fct<br>void cpu_6502_INC_abs(); | | | | | |

logic

```
M = (M + 1) & $FF
P.N = M.7
P.Z = (M==0) ? 1:0
```

comment

| | | | | | |
|---|---|---|---|---|---|
| op<br>0xF0 | name<br>BEQ | adr<br>rel | NVBDIZC<br>------- | | |
| example<br>BEQ $AB | | | | bytes<br>2 | cycles<br>2 |
| description<br>branch on equal (zero set) | | | | | |
| formal<br>branch on Z=1 | | | | | |
| c–fct<br>void cpu_6502_BEQ_rel(); | | | | | |

logic

```
if (P.Z == 1) GOTO (PC+M)
```

comment

| | | | | | |
|---|---|---|---|---|---|
| op<br>0xF1 | name<br>SBC | adr<br>izy | NVBDIZC<br>**---** | | |
| example<br>SBC ($A5),X | | | | bytes<br>2 | cycles<br>5 |
| description<br>subtract with carry | | | | | |
| formal<br>A:=A-adr | | | | | |
| c–fct<br>void cpu_6502_SBC_izy(); | | | | | |

logic

```
IF (P.D)
  t = bcd(A) - bcd(M) - !P.C
  P.V = (t>99 OR t<0) ? 1:0
ELSE
  t = A - M - !P.C
  P.V = (t>127 OR t<-128) ? 1:0
P.C = (t>=0) ? 1:0
P.N = t.7
P.Z = (t==0) ? 1:0
A = t & 0xFF
```

comment

| | | | | | |
|---|---|---|---|---|---|
| op<br>0xF5 | name<br>SBC | adr<br>zpx | NVBDIZC<br>**---** | | |
| example<br>SBC $AB,X | | | | bytes<br>2 | cycles<br>4 |
| description<br>subtract with carry | | | | | |
| formal<br>A:=A-adr | | | | | |
| c–fct<br>void cpu_6502_SBC_zpx(); | | | | | |

logic

```
IF (P.D)
  t = bcd(A) - bcd(M) - !P.C
  P.V = (t>99 OR t<0) ? 1:0
ELSE
  t = A - M - !P.C
  P.V = (t>127 OR t<-128) ? 1:0
P.C = (t>=0) ? 1:0
P.N = t.7
P.Z = (t==0) ? 1:0
A = t & 0xFF
```

comment

| | | | | |
|---|---|---|---|---|
| op<br>0xF6 | name<br>INC | adr<br>zpx | NVBDIZC<br>*----*- | |
| example<br>INC $AB,X | | bytes<br>2 | cycles<br>6 | |
| description<br>increment | | | | |
| formal<br>adr:=adr+1 | | | | |
| c–fct<br>`void cpu_6502_INC_zpx();` | | | | |

logic
```
M = (M + 1) & $FF
P.N = M.7
P.Z = (M==0) ? 1:0
```

comment

| | | | | |
|---|---|---|---|---|
| op<br>0xF8 | name<br>SED | adr<br>imp | NVBDIZC<br>---1--- | |
| example<br>SED | | bytes<br>1 | cycles<br>2 | |
| description<br>set decimal | | | | |
| formal<br>D:=1 | | | | |
| c–fct<br>`void cpu_6502_SED_imp();` | | | | |

logic
```
P.D = 1
```

comment

| | | | | |
|---|---|---|---|---|
| op<br>0xF9 | name<br>SBC | adr<br>aby | NVBDIZC<br>**---** | |
| example<br>SBC $ABCD,Y | | bytes<br>3 | cycles<br>4 | |
| description<br>subtract with carry | | | | |
| formal<br>A:=A-adr | | | | |
| c–fct<br>`void cpu_6502_SBC_aby();` | | | | |

logic
```
IF (P.D)
  t = bcd(A) - bcd(M) - !P.C
  P.V = (t>99 OR t<0) ? 1:0
ELSE
  t = A - M - !P.C
  P.V = (t>127 OR t<-128) ? 1:0
P.C = (t>=0) ? 1:0
P.N = t.7
P.Z = (t==0) ? 1:0
A = t & 0xFF
```

comment

| | | | | |
|---|---|---|---|---|
| op<br>0xFD | name<br>SBC | adr<br>abx | NVBDIZC<br>**---** | |
| example<br>SBC $ABCD,X | | bytes<br>3 | cycles<br>4 | |
| description<br>subtract with carry | | | | |
| formal<br>A:=A-adr | | | | |
| c–fct<br>`void cpu_6502_SBC_abx();` | | | | |

logic
```
IF (P.D)
  t = bcd(A) - bcd(M) - !P.C
  P.V = (t>99 OR t<0) ? 1:0
ELSE
  t = A - M - !P.C
  P.V = (t>127 OR t<-128) ? 1:0
P.C = (t>=0) ? 1:0
P.N = t.7
P.Z = (t==0) ? 1:0
A = t & 0xFF
```

comment

<table>
<tr><td>op<br>0xFE</td><td>name<br>INC</td><td>adr<br>abx</td><td colspan="2">NVBDIZC<br>*----*-</td><td>logic<br>M = (M + 1) &amp; $FF<br>P.N = M.7<br>P.Z = (M==0) ? 1:0</td></tr>
<tr><td colspan="3">example<br>INC $ABCD,X</td><td>bytes<br>3</td><td>cycles<br>7</td><td rowspan="4">comment</td></tr>
<tr><td colspan="5">description<br>increment</td></tr>
<tr><td colspan="5">formal<br>adr:=adr+1</td></tr>
<tr><td colspan="5">c–fct<br>void cpu_6502_INC_abx();</td></tr>
</table>

| Text1 | Text2 | Text3 | Text1 | Text1 | Text1 | | | |
|---|---|---|---|---|---|---|---|---|
| opcode | mnmonic | bytes | cycles | adr"-mode | function | flags | logic | example |
| 0x00 | BRK | 1 | 0 | imp | interrupt<br>(S)-=:PC,P<br>PC:=($FFFE) | NVBDIZC<br>-1-1- | bPoke(SP,PC.h)<br>SP = SP - 1<br>bPoke(SP,PC.l)<br>SP = SP - 1<br>bPoke(SP, (P\|\$10) )<br>SP = SP - 1<br>l = bPeek(\$FFFE)<br>h = bPeek(\$FFFF)<<8<br>PC = h\|l | BRK |
| 0x01 | ORA | 2 | 6 | izx | or with accumulator<br>A:=A or adr | NVBDIZC<br>*--*- | P.N = A.7<br>P.Z = (A==0) ? 1:0 | ORA ($A5,X) |
| 0x05 | ORA | 2 | 3 | zp | or with accumulator<br>A:=A or adr | NVBDIZC<br>*--*- | P.N = A.7<br>P.Z = (A==0) ? 1:0 | ORA $AB |

Table 2: Caption