```
GeeBee's GB Assembly Code Tips v1.0
-----------------------------------

Note: The following references to 'cycles' refers
 to machine cycles. To convert to clock cycles
 multiply by 4. (i.e. 1 machine cycle = 4 clock cycles)


; **** Load A with $00 ****
Method 1:
  ld a,0        ; 2 bytes, 2 cycles, Doesn't affect flags

Method 2:
  xor a         ; 1 byte, 1 cycle, Flag results: C=0, Z=1


; **** Compare A to $00 ****
Method 1:
  cp 0          ; 2 bytes, 2 cycles

Method 2:
  or a          ; 1 byte, 1 cycle

Method 3:
  and a         ; 1 byte, 1 cycle


; **** Call/Return ****
Method 1:       ; 4 bytes, 10 cycles
  ...
  call sub
  ret

Method 2:       ; 3 bytes, 4 cycles
  ...
  jp sub


; **** Exchange DE & HL ****
Method 1:        ; 6 bytes, 6 cycles
  ld a,d
  ld d,h
  ld h,a
  ld a,e
  ld e,l
  ld l,a

Method 2:       ; 4 bytes, 9 cycles
  push de
  ld d,h
  ld e,l
  pop  hl


; **** Load hl,[Address] ****
Method 1:
  ld a,[Address]        ; 8 bytes, 10 cycles
  ld l,a
  ld a,[Address+1]
  ld h,a

Method 2:
  ld hl,Address         ; 6 bytes, 8 cycles
```

```
  ld a,[hl+]
  ld h,[hl]
  ld l,a


; **** Call [HL] ****
Method 1:               ; 5 bytes, 8 cycles
  ld de,.retadr
  push de
  jp [hl]
.retadr:

Method 2:               ; 4 bytes, 7 cycles
  call DoJump
  ...
  ...

DoJump:
  jp [hl]


; **** HL = -HL ****
Method 1:               ; 7 bytes, 8 cycles
  ld a,l
  cpl
  ld l,a
  ld a,h
  cpl
  ld h,a
  inc hl

Method 2:               ; 7 bytes, 7 cycles
  xor a
  sub l
  ld l,a
  ld a,0
  sbc h
  ld h,a


; **** A = CONST - A ****
Method 1:               ; 4 bytes, 4 cycles
  ld b,a
  ld a,CONST
  sub b

Method 2:               ; 3 bytes, 3 cycles
  cpl
  add CONST+1


; **** HL = HL + A ****
Method 1:               ; 6 bytes, 6 cycles
  add l
  ld l,a
  ld a,0
  adc h
  ld h,a

Method 2:               ; 5 bytes, 5 cycles
  add l
  ld l,a
  jr nc,.notcarry
  inc h
.notcarry:
```

```
; **** Parameter Setup ****
Method 1:               ; 10 bytes

Entry1:
  ld a,1
  jr Sub
Entry2:
  ld a,2
  jr Sub
Entry3:
  ld a,3
Sub:
  ...
  ...

Method 2:               ; 8 bytes

Entry1:
  ld a,1
  DB 1     ; Opcode for LD BC,xxxx
Entry2:
  ld a,2
  DB 1     ; Opcode for LD BC,xxxx
Entry3:
  ld a,3
Sub:
  ...
  ...


; **** Fast subroutine execution ***

Method 1:
  ld hl,param1
  call sub1
  ld hl,param2
  call sub2
  ld hl,param3
  call sub1
  ...
  ...

.sub1:
  ...
  ret
.sub2:
  ...
  ret

Method 2:
  ld sp,calltable
  ret           ; jump to sub1

.sub1:
  pop hl
  ...
  ret
.sub2:
  pop hl
  ...
  ret

calltable:
```

```
  dw sub1,param1
  dw sub2,param2
  dw sub1,param3

** End of File ***
```

```
Jeff's GB Assembly Code Tips v1.0
---------------------------------

Note: The following references to 'cycles' refers
 to machine cycles. To convert to clock cycles
 multiply by 4. (i.e. 1 machine cycle = 4 clock cycles)

; **** Load A with $00 ****
Method 1:
  ld a,0        ; 2 bytes, 2 cycles, Doesn't affect flags

Method 2:
  xor a         ; 1 byte, 1 cycle, Flag results: C=0, Z=1

; **** Compare A to $00 ****
Method 1:
  cp 0          ; 2 bytes, 2 cycles

Method 2:
  or a          ; 1 byte, 1 cycle

Method 3:
  and a         ; 1 byte, 1 cycle

; **** Call/Return ****
Method 1:       ; 4 bytes, 10 cycles
  ...
  call sub
  ret

Method 2:       ; 3 bytes, 4 cycles
  ...
  jp sub

; **** Exchange DE & HL ****

Method 1:        ; 6 bytes, 6 cycles
  ld a,d
  ld d,h
  ld h,a
  ld a,e
  ld e,l
  ld l,a

Method 2:       ; 4 bytes, 9 cycles
  push de
  ld d,h
  ld e,l
```

```
    pop  hl

  ; **** Load hl,[Address] ****

  Method 1:
    ld a,[Address]        ; 8 bytes, 10 cycles
    ld l,a
    ld a,[Address+1]
    ld h,a

  Method 2:
    ld hl,Address         ; 6 bytes, 8 cycles
    ld a,[hl+]
    ld h,[hl]
    ld l,a

  ; **** Call [HL] ****

  Method 1:               ; 5 bytes, 8 cycles

    ld de,.retadr

    push de

    jp [hl]

  .retadr:



  Method 2:               ; 4 bytes, 7 cycles

    call DoJump

    ...

    ...


  DoJump:

    jp [hl]




  ; **** HL = -HL ****

  Method 1:               ; 7 bytes, 8 cycles

    ld a,l

    cpl

    ld l,a

    ld a,h

    cpl

    ld h,a

    inc hl
```

```
Method 2:               ; 7 bytes, 7 cycles
  xor a
  sub l
  ld l,a
  ld a,0
  sbc h
  ld h,a


; **** A = CONST - A ****
Method 1:               ; 4 bytes, 4 cycles
  ld b,a
  ld a,CONST
  sub b

Method 2:               ; 3 bytes, 3 cycles
  cpl
  add CONST+1


; **** HL = HL + A ****
Method 1:               ; 6 bytes, 6 cycles
  add l
  ld l,a
  ld a,0
  adc h
  ld h,a

Method 2:               ; 5 bytes, 5 cycles
  add l
  ld l,a
```

```
    jr nc,.notcarry
    inc h
  .notcarry:



  ; **** Parameter Setup ****
  Method 1:               ; 10 bytes

  Entry1:
    ld a,1
    jr Sub
  Entry2:
    ld a,2
    jr Sub
  Entry3:
    ld a,3
  Sub:
    ...
    ...

  Method 2:               ; 8 bytes

  Entry1:
    ld a,1
    DB 1     ; Opcode for LD BC,xxxx
  Entry2:
    ld a,2
    DB 1     ; Opcode for LD BC,xxxx
  Entry3:
    ld a,3
  Sub:
    ...
    ...
```

```
; **** Fast subroutine execution ***

Method 1:
  ld hl,param1
  call sub1
  ld hl,param2
  call sub2
  ld hl,param3
  call sub1
  ...
  ...

.sub1:
  ...
  ret
.sub2:
  ...
  ret

Method 2:
  ld sp,calltable
  ret           ; jump to sub1

.sub1:
  pop hl
  ...
  ret
.sub2:
  pop hl
  ...
```

```
    ret


  calltable:

    dw sub1,param1

    dw sub2,param2

    dw sub1,param3


  ** End of File ***
```

```
  FF04            rDIV            Divider Register
  FF05            rTIMA           Timer Counter
  FF06            rTMA            Timer Modulo
  FF07            rTAC            Timer Control

  FF0F            rIF             Interrupt Flag
  FF10            rNR10           Sound Channel 1 : Sweep
  FF11            rNR11           Sound Channel 1 : Length / Duty
  FF12            rNR12           Sound Channel 1 : Envelope
  FF13            rNR13           Sound Channel 1 : Frequency Lo
  FF14            rNR14           Sound Channel 1 : Frequency Hi

  FF16            rNR21           Sound Channel 2 : Length / Duty
  FF17            rNR22           Sound Channel 2 : Envelope
  FF18            rNR23           Sound Channel 2 : Frequency Lo
  FF19            rNR24           Sound Channel 2 : Frequency Hi

  FF1A            rNR30           Sound Channel 3 : On / Off
  FF1B            rNR31           Sound Channel 3 : Sound Length
  FF1C            rNR32           Sound Channel 3 : Output Level
  FF1D            rNR33           Sound Channel 3 : Frequency Lo
  FF1E            rNR34           Sound Chennel 3 : Frequency Hi

  FF20            rNR41           Sound Channel 3 : Sound Length
  FF21            rNR42           Sound Channel 3 : Envelope
  FF22            rNR42_2         Sound Channel 3 : Polynomial Counter
  FF23            rNR43           Sound Channel 3 : ??
  FF24            rNR50           Channel Control
  FF25            rNR51           Sound Output Terminal Selection
  FF26            rNR52           Sound On / Off
```