switch-Anweisung

Syntax

```
Statement =  ...
   |  "switch" "(" Expr ")" "{" Case { Case} "}"
   |  ... .
Case        = "case" number ":" { Statement }.
```

Beispiel

```
switch (month) {
                case 1: case 3: case 5: case 7:
                case 8: case 10: case 12:
                                days = 31; break;
                case 4: case 6: case 9: case 11:
                                days = 30; break;
                case 2:
                                days = 28;
}
```

Bedeutung

- Es wird zu der durch *Expr* bezeichneten *Case*-Marke gesprungen.
- Wenn *Expr* zu keiner *Case*-Marke passt, wird die switch-Anweisung übersprungen.
- Am Ende jedes *Case*-Zweigs sollte ein Aussprung mittels *break* erfolgen, ansonsten läuft das Programm in den nächsten *Case*-Zweig.

Kontextbedingungen

```
Statement =  "switch" "(" Expr ")" "{" Case { Case} "}".
```

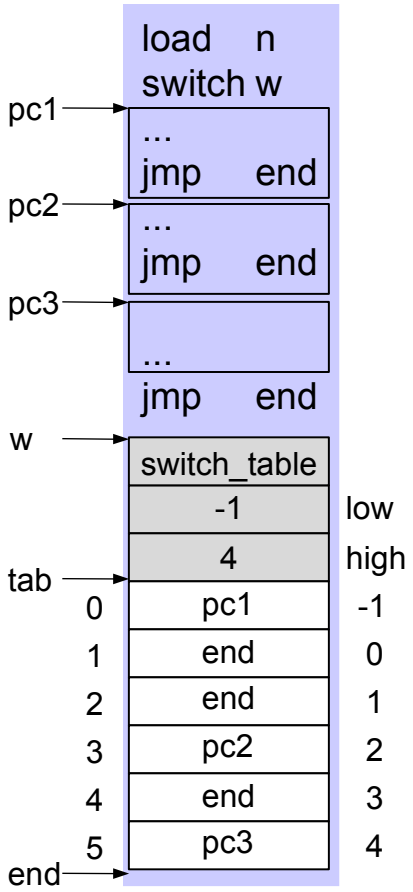
- *Expr* muss vom Typ *int* sein.
- Die Differenz zwischen kleinster und größter Case-Marke muss ≤ 100 sein.

Behandlung in der VM

Bytecode

switch w	..., n
	...
		Switch
		n = pop(); tab = w + 1 + 4 + 4;
		low = tab[-2]; high = tab[-1];
		if (low <= n && n <= high)
		pc = tab[n - low];
		else
		pc = tab + (high - low + 1);

```
switch (i) {
                case -1:    ...; break;
                case  2:    ...; break;
                case  4:    ...;
}
```



Hinweise für die Codeerzeugung

Aufgaben des Compilers

```
switch (i) {
                case -1:    ...; break;
                case  2:    ...; break;
                case  4:    ...;
}
```

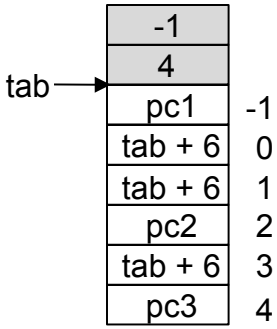
- baut Liste von Paaren (*Marke*, *PC*) auf

-1	pc1
2	pc2
4	pc3

- berechnet daraus *low* und *high*

  low = -1, *high* = 4

- gibt am Ende der switch-Anweisung aus:
  - jmp nach *end*
  - switch_table, *low*, *high* und Tabelle



- patcht switch-Instruktion und Vorwärtssprünge zu *end*