1. ADD(R)

|  |  |
| --- | --- |
| PC → Mem\_a/ alu1\_a  Mem\_d → ir  +1 → alu1\_b  alu1\_out → PC | ADD |
|  |
| no mod c and z flag |
| S1: HKT |
|  |
| ir9-11 → rf\_a1  ir6-8 →rf\_a2  rf\_d1 → t1  rf\_d2 → t2 | S2: OR |
| if((Op = ADC or NDC) and (flagc = 0)) then PC -> R7 |
| if((Op = ADZ or NDZ) and (flagz = 0)) then PC -> R7 |
|  |
|  |
| t1 → alu2\_a  t2 → alu2\_b  alu2\_out →t1 | ADD/NAND(OP code) |
| mod c (only if ADD) and z flag |
| S3: ALU\_op |
| If (op is add,adc,adz,ndu,ndc,ndz) load t2 in alu\_b and operation in ir and modify corresponding flags(S3) |
| elsif(op is adi) ir0-5 → SE6 → alu\_b and add and modify corresponding flags(S3’) |
| elsif(op is lw) t2 -> alu\_a (and not t1) and ir0-5 → SE6 → alu\_b and add and do not modify any flags (S3’’) |
| elsif(op is sw) t2 -> alu\_a (and not t1) and ir0-5 → SE6 → alu\_b and alu\_out to t2 instead of t1 in all previous cases and add and do not modify any flags (S3’’) |
| t1 → rf\_d3  ir3-5 → rf\_a3  if(rf\_a3 == R7){  t1->PC  }  else{  PC-> R7} | rf write |
| S4 |
| If(op is adc adz add ndu ndz ndc) ir3-5 -> rf\_a3(S4) |
| elsif(op is adi) ir6-8 -> rf\_a3 (S4’) |
| elsif(op is lhi) ir0-8 → LSHIFT-7 → rf\_d3  ir9-11 → rf\_a3 and ir0-8 → LSHIFT-7 → PC  (after the change in red, no t2 -> rf\_d3 found, replace the signal from datapath and control signals : VERIFY) (S4’’) |
| check rf\_a3 for 111 and load PC also |

1. ADC(R)

|  |  |
| --- | --- |
| PC → Mem\_a/ alu1\_a  Mem\_d → ir  +1 → alu1\_b  alu1\_out → PC |  |
|  |
|  |
| S1: HKT |
|  |
| ir9-11 → rf\_a1  ir6-8 →rf\_a2  rf\_d1 → t1  rf\_d2 → t2  If (C flag==0 & OPCODE) then PC -> R7 |  |
|  |
| S2 |
| for ADC,,NDC if(C==0 & CZ = 10),  for ADZ, NDZ if(Z==0 & CZ ==01)  goto S1 |
|  |
| if(C==1){enable t1, CZ}  else (disable the t1, CZ)  t1 → alu2\_a  t2 → alu2\_b  alu2\_out →t1 | ADD/NAND(OP code) |
|  |
| S3:ALU\_op |
|  |
|  |
| t1 → rf\_d3  ir3-5 → rf\_a3  if(rf\_a3 == R7){  t1->PC  }  else{  PC-> R7} |  |
| S4 |
|  |
|  |
|  |

1. ADZ(R)

|  |  |
| --- | --- |
| PC → Mem\_a/ alu1\_a  Mem\_d → ir  +1 → alu1\_b  alu1\_out → PC |  |
|  |
|  |
| S1: HKT |
|  |
| ir9-11 → rf\_a1  ir6-8 →rf\_a2  rf\_d1 → t1  rf\_d2 → t2  If (Z flag==0 & OPCODE) then PC -> R7 |  |
|  |
| S2 |
| for ADC,,NDC if(C==0 & CZ = 10),  for ADZ, NDZ if(Z==0 & CZ ==01)  goto S1 |
|  |
| if(Z==1){enable t1, CZ}  else (disable the t1, CZ)  t1 → alu2\_a  t2 → alu2\_b  alu2\_out →t1 | ADD/NAND(OP code) |
| S3:ALU\_op |
|  |
|  |
|  |
| t1 → rf\_d3  ir3-5 → rf\_a3  if(rf\_a3 == R7){  t1->PC  }  else{  PC-> R7} |  |
|  |
| S4 |
|  |
|  |

1. ADI (I)

|  |  |
| --- | --- |
| PC → Mem\_a/ alu1\_a  Mem\_d → ir  +1 → alu1\_b  alu1\_out → PC |  |
|  |
|  |
| S1: HKT |
|  |
| ir9-11 → rf\_a1  ir6-8 →rf\_a2  rf\_d1 → t1  rf\_d2 → t2 |  |
|  |
|  |
| S2 |
|  |
| t1 → alu2\_a  ir0-5 → SE6 → alu2\_b  alu2\_out →t1 |  |
|  |
| S3’ |
|  |
|  |
| t1 → rf\_d3  ir6-8 → rf\_a3  if(rf\_a3 == R7){  t1->PC  }  else{  PC-> R7} |  |
|  |
| S4’ |
|  |
|  |

1. NDU
2. NDC
3. NDZ
4. LHI (J)

|  |  |
| --- | --- |
| PC → Mem\_a/ alu1\_a  Mem\_d → ir  +1 → alu1\_b  alu1\_out → PC |  |
|  |
|  |
| S1: HKT |
|  |
| ir9-11 → rf\_a1  ir6-8 →rf\_a2  rf\_d1 → t1  rf\_d2 → t2 |  |
|  |
| S2 |
|  |
|  |
| ir0-8 → LSHIFT-7→ t2 |  |
|  |
| S6 |
|  |
|  |
| ir0-8 → LSHIFT-7 → rf\_d3  ir9-11 → rf\_a3  if(rf\_a3 == R7){  ir0-8 → LSHIFT-7->PC  }  else{  PC-> R7} |  |
|  |
| S4’’ |
|  |
| DECODER change (NEXT STATE LOGIC CHANGE) |

1. LW (I)

|  |  |
| --- | --- |
| PC → Mem\_a/ alu1\_a  Mem\_d → ir  +1 → alu1\_b  alu1\_out → PC |  |
|  |
| S1: HKT |
|  |
|  |
| ir9-11 → rf\_a1  ir6-8 →rf\_a2  rf\_d1 → t1  rf\_d2 → t2 |  |
|  |
| S2 |
|  |
|  |
| ir0-5 → SE6→ alu2\_B  t2 → alu2\_a  alu2\_out → t1 |  |
| S3’’ |
| Disable CZ |
|  |
|  |
| t1 → Mem\_a  Mem\_d → t1 |  |
|  |
| S7 |
|  |
|  |
| t1 → alu2\_a, rf\_d3  0 → alu\_b  ir9-11 → rf\_a3  if(rf\_a3 == R7){  t1->PC  }  else{  PC-> R7} | ADD |
| Disable only C |
| S8 |
|  |
|  |

1. SW

|  |  |
| --- | --- |
| PC → Mem\_a/ alu1\_a  Mem\_d → ir  +1 → alu1\_b  alu1\_out → PC |  |
|  |
|  |
| S1: HKT |
|  |
| ir9-11 → rf\_a1  ir6-8 →rf\_a2  rf\_d1 → t1  rf\_d2 → t2 |  |
|  |
| S2 |
|  |
|  |
| ir0-5 → SE6→ alu2\_b  t2 → alu2\_a  alu2\_out → t2 |  |
|  |
| Disable CZ |
| S3’’ |
|  |
| t1 → Mem\_d  t2 → Mem\_a  111 → rf\_a3  PC → R7 |  |
| WR |
| S9 |
|  |
|  |

1. LM

|  |  |
| --- | --- |
| PC → Mem\_a/ alu1\_a  Mem\_d → ir  +1 → alu1\_b  alu1\_out → PC |  |
|  |
|  |
| S1: HKT |
|  |
| ir9-11 → rf\_a1  ir6-8 →rf\_a2  rf\_d1 → t1  rf\_d2 → t2 |  |
|  |
| S2 |
|  |
|  |
| ir0-8 → SE9 → t2 |  |
| (no S6 after the change in red in lhi flowchart) |
| S6’ |
|  |
|  |
| t2 → PE (PE clears least significant 1 of t2)  PE -> t2  PE\_addr → t4 (3 bit)  t1 → Mem\_a  Mem\_d → t3 | See if PE has to detect and clear the first least significant 1 or most significant 1 |
|  |
|  |
| S10: |
|  |
| t4 → rf\_a3  t3 → rf\_d3  t1 → alu1\_a  +1 → alu1\_b  alu1\_out → t1  if(rf\_a3 == R7){  t3->PC  }  else{  PC-> R7} | (Disable t2 and t4 write) |
| if( t2=0x0000) goto S1 (HKT) |
| loopback to S10 |
| S11: |
| If (t4 == 111){  t3 -> PC}  else{ PC -> R7}  (DO NOT load rf\_a3 with 111 while doing PC to R7) |

1. SM

|  |  |
| --- | --- |
| PC → Mem\_a/ alu1\_a  Mem\_d → ir  +1 → alu1\_b  alu1\_out → PC |  |
|  |
|  |
| S1: HKT |
|  |
| ir9-11 → rf\_a1  ir6-8 →rf\_a2  rf\_d1 → t1  rf\_d2 → t2 |  |
|  |
| S2 |
|  |
|  |
| ir0-8 → SE9 → t2 |  |
|  |
| S6’ |
|  |
|  |
| t2 → PE (PE clears least significant 1 of t2)  PE\_addr → t4 (3 bit) | enable t2 write from PE |
| See if PE has to detect and clear the first least significant 1 or most significant 1 |
|  |
| S12: |
|  |
| t4 → rf\_a1  rf\_d1 → t3 |  |
| Disable T2and t4 write |
| S13 |
|  |
|  |
| t1 → Mem\_a  t3 → Mem\_d  t1 → alu1\_a  +1 → alu1\_b  alu1\_out → t1  PC -> R7 | (Disable t2 and t4 write) |
| if( t2=0x0000) goto S1 (HKT) |
| else loopback to S12 |
| S14 |
| LOAD rf\_a3 with 111 while doing PC -> R7 |

1. BEQ

|  |  |
| --- | --- |
| PC → Mem\_a/ alu1\_a  Mem\_d → ir  +1 → alu1\_b  alu1\_out → PC |  |
|  |
|  |
| S1: HKT |
|  |
| ir9-11 → rf\_a1  ir6-8 →rf\_a2  rf\_d1 → t1  rf\_d2 → t2 |  |
|  |
| S2 |
|  |
|  |
| t1 → alu2\_a  t2 → alu2\_b  111 → rf\_a1  rf\_d1 → t1 | subtract (or compare) |
| no modify C and Z flags |
| S15 (alu-z -> tempz has to be done everywhere except S16) |
|  |
| if (tempz==0){  PC -> R7  }  else{  t1 → alu2\_a  ir0-8 → SE9 → alu2\_b  alu\_out → PC,R7  }  else{  t1 → t2  } | add(tempz\_en disable) |
| no modify c and z flags |
| S16 |
| put rf\_a3 as 111 while choosing alu\_out in r7write mux |
| next state is S1  DECODER change, NEXT STATE LOGIC CHANGE) |

1. JAL

|  |  |
| --- | --- |
| PC → Mem\_a/ alu1\_a  Mem\_d → ir  +1 → alu1\_b  alu1\_out → PC |  |
|  |
|  |
| S1: HKT |
|  |
| ir9-11 → rf\_a1  ir6-8 →rf\_a2  rf\_d1 → t1  rf\_d2 → t2 |  |
|  |
| S2 |
|  |
|  |
| 111 → rf\_a1  rf\_d1 → t3 |  |
|  |
| S17 |
|  |
|  |
| ir0-8 → SE9 → alu2\_a  t3 → alu2\_b, rf\_d3  alu\_out → PC,R7  ir9-11 → rf\_a3 | add (no flags modified) |
| choose alu--out from r7 write mux |
| S18 |
|  |
|  |

1. JLR

|  |  |
| --- | --- |
| PC → Mem\_a/ alu1\_a  Mem\_d → ir  +1 → alu1\_b  alu1\_out → PC |  |
|  |
|  |
| S1: HKT |
|  |
| ir9-11 → rf\_a1  ir6-8 →rf\_a2  rf\_d1 → t1  rf\_d2 → t2 |  |
|  |
| S2 |
|  |
|  |
| 111 → rf\_a1  rf\_d1 → t3 |  |
| S17 |
|  |
|  |
|  |
| t3 → rf\_d3  t2 → PC,R7  ir9-11 → rf\_a3 |  |
|  |
| S19 |
|  |
|  |